ПЕРМСКИЙ
ГОСУДАРСТВЕННЫЙ
НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

АКТУАЛЬНЫЕ ПРОБЛЕМЫ МАТЕМАТИКИ, МЕХАНИКИ И ИНФОРМАТИКИ 2021

Сборник статей по материалам студенческой конференции (г. Пермь, 25 мая – 10 июня 2021 г.)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

АКТУАЛЬНЫЕ ПРОБЛЕМЫ МАТЕМАТИКИ, МЕХАНИКИ И ИНФОРМАТИКИ 2021

Сборник статей по материалам студенческой конференции
(г. Пермь, 25 мая – 10 июня 2021 г.)

Пермь 2021

УДК 51+531+004.8](082)
ББК 22+32.81
А437

Актуальные проблемы математики, механики и информатики 2021 [Электронный ресурс] : сборник статей по материалам студенческой конференции (г. Пермь, 25 мая – 10 июня 2021 г.) / под редакцией А. П. Шкарапута ; Пермский государственный национальный исследовательский университет. – Электронные данные. – Пермь, 2021. – 9,27 Мб ; 217 с. – Режим доступа: http://www.psu.ru/files/docs/science/books/sborniki/aktualnye-problemy-matematiki-mekhaniki-informatiki-2021.pdf. – Заглавие с экрана.

ISBN 978-5-7944-3665-5

В сборнике представлены материалы студенческой конференции Актуальные проблемы математики, механики и информатики 2021, которая проводилась 25 мая – 10 июня 2021 г. в г. Перми.

Сборник предназначен для научных и педагогических работников, преподавателей, аспирантов, магистрантов, студентов и всех, кто интересуется проблемами математики, механики и информатики.

УДК 51+531+004.8](082)
ББК 22+32.81

Издается по решениюученого совета механико-математического факультета Пермского государственного национального исследовательского университета

ISBN 978-5-7944-3665-5

ОГЛАВЛЕНИЕ

УДК 65.011.56

ОЦЕНКА ЭФФЕКТИВНОСТИ ВНЕДРЕНИЯ BSS В ТЕЛЕКОММУНИКАЦИОННОЙ КОМПАНИИ

Базанова Мария Валерьевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, bazanova.mariya.v@gmail.com

BSS (Business Support System – система поддержки бизнеса) предназначена для комплексного управления ресурсами предприятия. BSS системы содержат множество модулей и подсистем, сочетание которых с корпоративными информационными системами обеспечивает необходимую функциональность для решения различных вопросов. BSS оптимизирует все ключевые процессы компании, уменьшает сложность процессов, сокращает число ненужных действий, понижает риск возникновения ошибок и благотворно влияет на расходы. Проблема, с которой часто сталкиваются компании, заключается в трудностях измерения эффективности инвестиций в ИТ, согласовании с бизнес-целями и выгодами для затрагиваемой части организации. Определяемые критерии успешного внедрения системы должны позволять количественно оценивать степень удовлетворения каждой из потребностей, связанных с внедрением.

Ключевые слова: BSS, модули BSS, эффективность внедрения, KPI, TCO, ROI, CBA.

Вся деятельность телекоммуникационной компании строится на различных бизнес-процессах, для обеспечения каждого из которых может использоваться отдельный программный продукт от конкретного вендора. В этом случае неизбежно возникают сложности с интеграцией. Данные из одной программы должны быстро и без потерь передаваться в другую. Но если инструменты выпущены разными производителями, это становится сложно организовать. Возникает необходимость в доработке и оптимизации систем. Также развитие бизнеса требует развития информационной системы, часто это решается добавлением очередного программного продукта. Так проблемы накапливаются и держат в постоянном напряжении ИТ-департамент, увеличивают финансовые затраты на управление этими системами.

Гораздо более простой и оптимальный путь – использовать полный комплекс программных продуктов BSS от одного вендора, который уже позаботился о том, чтобы отдельные модули взаимодействовали максимально эффективно и быстро. BSS – это аббревиатура от английского Business Support System (система поддержки бизнеса). Это комплекс программных продуктов, который отвечает за учет услуг, состояние счета и т.д. Основа BSS – биллинг и CRM. BSS оптимизирует все ключевые процессы компании. Можно сказать, что BSS – это «сердце» управления бизнесом для всей компании. Это уменьшает сложность процессов, сокращает число ненужных действий, понижает риск возникновения ошибок и благотворно влияет на расходы. Помимо того, что сокращаются затраты на поддержку, развертывание и обучение персонала [1]. Конечно, это довольно серьезный проект, имеющий при этом и свои риски. Так мы видим, что трансформация BSS влияет как на клиентский опыт, на работу сотрудников и требует изменения текущих бизнес-процессов, что ложится на плечи аналитиков, бизнес-технологов и специалистов по методологии.

BSS предназначена для комплексного управления телекоммуникационными ресурсами предприятия. Эта аббревиатура объединяет несколько подклассов систем, включая мониторинг (Fault Management), сбор и анализ производительности (Performance Management), медиацию (сбор данных от разнородного оборудования и приведение их к

общему виду), SIEM (Security Information and Event Management) – сбор и обработка данных от средств информационной безопасности и ряд других систем [2].

Также одним из наиболее важных классов BSS системы является Fraud Management, что дословно переводится как «управление мошенничеством». Модуль Fraud Management, предназначенный, в первую очередь, для операторов связи, обеспечивает обнаружение, пресечение и предотвращение случаев несанкционированного доступа к ресурсам оператора. Оснащенная средствами мониторинга для различных типов соединений, система реагирует в случае вызова подозрительного номера, несуществующего пользователя или несанкционированного доступа к услугам. Эксперты также отмечают, что тесная интеграция Fraud Management с CRM-решением позволяет максимально оперативно и эффективно построить защиту от мошенничества.

Кроме класса Fraud Management, очень большое значение имеет модуль Fault Management & Trouble Ticketing – регистрация и управление неисправностями. Решение позволяет эффективно управлять планами работ, а также оптимизировать работу персонала. Сокращение сроков ремонтных работ, которое достигается при его внедрении, позволяет компании работать значительно более оперативно. Принцип действия Trouble Ticketing схож с Fraud Management: собирается и систематизируется информация обо всех возникающих проблемах и неполадках, кроме того, сохраняются данные о способе их устранения и текущем состоянии работ.

Помимо перечисленных классов, в современную BSS систему входит множество других модулей. Это и решения для управления инвентаризацией (Inventory Management), позволяющие автоматизировать планирование пополнения запасов и обеспечить наглядность, строгий контроль и учет одноименных ресурсов телекоммуникационной компании, решения для управления производительностью (Performance Management), предназначенные для оптимизации работы телекоммуникационной сети. Кроме того, в BSS решениях имеются модули для управления заказами (Order Management), а также аналитические классы для планирования и развития услуг (Network & Service Provisioning Management) и широко известные WorkFlow-системы, предназначенные для управления территориально-распределенными командами сотрудников. Средства WorkFlow Management обеспечивают также мониторинг и составление аналитических отчетов в режиме реального времени.

Можно выделить несколько возможных способов построения BSS решения на предприятии. Каждый из этих вариантов сводится в итоге к интеграции различных классов BSS с другими информационными системами и/или классами. Это может быть биллинговая система (автоматизированная система учёта предоставленных услуг, их тарификации и выставления счетов для оплаты) + CRM (Customer Relationship Management, управление отношениями с клиентами) + Fraud Management (система для борьбы с мошенничеством на сети связи) или CRM + SLA management (Service Level Agreement, соглашение об уровне предоставления услуги) + Fault management&Trouble ticketing (регистрация и управление неисправностями), а также некоторые другие сочетания [3]. Каждая комбинация обеспечивает решение определенного класса наиболее критичных для заказчика бизнес-задач. Выбор делается на основе комплексного анализа всех бизнес-процессов компании.

Компании необходимо провести более глубокую оценку, чтобы выяснить, могут ли вложения в информационные системы принести пользу или даже привести к убыткам для компании. Сегодня информационные системы – это не просто инструмент для работы, а инструмент, способствующий развитию бизнеса. Компании тратят бюджеты на инвестиции в информационные системы, чтобы удовлетворить потребности бизнеса, которые растут с каждым годом. Чтобы гарантировать, что информационные системы создают ценность для компании, инвестиции в них должны управляться и контролироваться. Ценность внедрения информационных систем не только материальна, но и нематериальна. Измерение нематериальных выгод сложнее, чем ощутимых выгод. Многие организации, которые оценивают инвестиции в информационные системы, основываются только на вещах, которые можно измерить напрямую, таких как окупаемость инвестиций (ROI), экономия

средств, эффективность времени и т.д. Трудности в оценке выгод являются основной причиной неопределенности в отношении ожидаемых выгод от инвестиций в информационные системы.

Организации, стремящиеся к положительной взаимосвязи между производительностью организации и инвестициями в информационные системы, часто игнорируют оценки инвестиций в информационные системы, поскольку они нематериальны и сложны. Одна из отраслей, где по-прежнему инвестируют средства во внедрение новых систем, – это телекоммуникационные компании. Информационные системы используются для удовлетворения операционных и стратегических потребностей. Это важный фактор, который может повысить эффективность бизнеса и стать одним из факторов, способствующих достижению видения, миссии, целей, ключевых показателей эффективности (KPI) и программ организации в качестве их ролей и обязанностей.

Проблема, с которой часто сталкивается организация, – это сложность измерения эффективности инвестиций, их соответствия бизнес-целям и выгод в той части организации, на которую они влияют. Определение выгод от инвестиций в информационные системы, связанных с функциями организации, может дать более полное представление о том, какая часть подразделения организации получает прямую выгоду.

Определяемые критерии успешного внедрения системы должны позволять количественно оценивать степень удовлетворения каждой из потребностей, связанных с внедрением. Кроме того, по каждому критерию должно быть определено его конкретное оптимальное значение [3]. Каждая компания имеет уникальную целевую структуру, которая отражает ее деление на ключевые функциональные области. У компании, связанной с оказанием услуг, структура ключевых областей может выглядеть следующим образом: маркетинговая деятельность, разработка стратегии, продвижение, продажи, обслуживание клиентов [4].

Сегодня наиболее часто используется система ключевых показателей эффективности – KPI (англ. key performance indicators). Они позволяют видеть бизнес в целом, не погружаясь в детали (в принципы начисления прибылей и убытков, движения денежных средств). При использовании системы KPI руководитель может всего на одной странице увидеть систему индикаторов, свидетельствующих о развитии и состоянии его компании [5].

Так как цифровые системы являются неотъемлемой частью бизнеса, они становятся объектом применения общеизвестных способов оценивания эффективности в отношении реализуемых проектов и проходящих процессов. Самые популярные – TCO и ROI.

ROI (Return of Investment) – это коэффициент возврата инвестиций. Методика применяется повсеместно при расчёте эффективности инвестиций, а также оценивания проектов в разных отраслях деятельности. ROI рассчитывается в качестве совокупной прибыли по годам, приведенной к первоначальному моменту времени и поделенной на величину инвестиций. В итоге определяется момент времени, при котором прибыль от внедрения покроет стоимость внедрения, то есть речь идёт об определении срока окупаемости [4].

Анализ эффективности затрат (CBA – Cost-benefit analysis) используют для оценки риска в ситуации, когда необходимо сравнить общие ожидаемые затраты с общими ожидаемыми выгодами (доходами и преимуществами) и выбрать лучший или наиболее выгодный вариант решения [3].

Полная стоимость владения (total cost of ownership, TCO) представляет собой универсальный термин, обозначающий совокупные затраты, с которыми связано использование продукта или услуги на протяжении всего их жизненного срока. Это методика расчета, создана чтобы помочь потребителям и руководителям предприятий определить прямые и косвенные затраты и выгоды, связанные с любым компонентом информационных систем. Цель ее применения – получить итоговую картину, которая отражала бы реальные затраты, связанные с приобретением определенных средств и технологий, и учитывала все аспекты их последующего использования [6].

Библиографический список

1. *Kangovi S.* Operations and Business Support Systems // Peering Carrier Ethernet Networks. 2017. С. 219 – 246.
2. Портал выбора технологий и поставщиков [Электронный ресурс] URL: https://www.tadviser.ru/index.php/Статья:OSS/BSS_системы (дата обращения: 20.04.2021).
3. *Nugroho W. S., Ranti B., Saputra D. A.* Benefits analysis of IT investment in Business Support System (BSS) projects using Ranti'sgeneric IS/IT business values: Case Studies of the Indonesian telecommunication company // 2019 International Conference on Advanced Computer Science and information Systems (ICACSIS). 2019. С. 331-336.
4. *Banu G. S.* Measuring innovation using key performance indicators // Procedia Manufacturing. 2018. №22. С. 906 – 911.
5. *Gu W. W. B., Tanaka Y., Yamori K.* Financial benefit analysis of macro-femto network structures based on TCO approach // 2016 18th International Conference on Advanced Communication Technology (ICACT). 2016. С. 838-842.
6. *Луценко И.Р.* Методика оценки эффективности внедрения ERP-систем автоматизации на предприятии // АНИ: экономика и управление. 2016. №2. С. 212-216.

EVALUATION OF THE EFFECTIVENESS OF IMPLEMENTING BSS IN A TELECOMMUNICATION COMPANY

Bazanova Mariya V.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, bazanova.mariya.v@gmail.com

BSS (Business Support System) is designed for complex enterprise resource management. BSS systems contain many modules and subsystems, the combination of which with corporate information systems provides the necessary functionality for solving various issues. BSS optimizes all key processes of the company, reduces the complexity of processes, reduces unnecessary steps and the risk of errors and has a beneficial effect on costs. The challenge that companies often face is the difficulty of measuring the effectiveness of IT investments, aligning with business goals and benefits for the affected part of the organization. The defined criteria for the successful implementation of the system should allow to quantify the degree of satisfaction of each of the needs associated with the implementation.

Key words: BSS, BSS modules, efficiency of implementation, KPI, TCO, ROI, CBA.

УДК 004.056.53

ИССЛЕДОВАНИЕ СТАТИСТИЧЕСКИХ ЗАВИСИМОСТЕЙ ПОРЯДКА ГРУППЫ ТОЧЕК ЭЛЛИПТИЧЕСКОЙ КРИВОЙ ОТ ЕЁ ПАРАМЕТРОВ

Бровкин Иван Алексеевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, bibika002@yandex.ru

Рассматриваются эллиптические кривые в форме Вейерштрасса. Описаны общие особенности алгебры в группе точек эллиптических кривых. Рассказано о применении эллиптических кривых в криптографии, особенно в российских стандартах. Исследованы способы формирования эллиптических кривых для использования в криптографии. Написана программа, создающая определенную выборку эллиптических кривых. Сформирована выборка эллиптических кривых, для них считается порядок группы точек. Данные проанализированы с помощью математических пакетов. Для данной вероятностной модели исследованы различные гипотезы зависимостей, которые могут влиять на распределение вероятностей простого порядка группы точек эллиптической кривой. Показано, что распределение простых порядков группы точек эллиптической кривой не зависит от параметров этой кривой, а также разностей этих параметров или отношений.

Ключевые слова: криптография, эллиптические кривые, теория групп.

В связи с повсеместным распространением средств передачи информации и информационных технологий в наши дни очень важным становится вопрос о защите данных, которые нужно передавать, хранить и обрабатывать. У пользователей информационных систем есть необходимость в надёжном способе передачи данных. Безопасность в свою очередь это способность обеспечить конфиденциальность и целостность. Данная потребность привела к бурному развитию криптографических средств защиты информации. Современные протоколы позволяют пользователям систем передачи данных быть абсолютно уверенными в надежности используемых систем.

Один из способов обеспечения безопасности является электронная подпись, то есть «информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию» [1].

Эллиптическая кривая – это множество точек (x, y), описывающихся уравнением:

$$y^2 = x^3 + ax + b, \quad (1.1)$$

где a, b – коэффициенты кривой. В приложении к криптографии эллиптическая кривая над конечным простым полем $GF(p)$ определяется как множество (x, y), таких что $x, y \equiv GF(p)$, удовлетворяющих уравнению:

$$y^2 = x^3 + ax + b \ (mod\ p), \quad (1.2)$$

где $a, b \equiv GF(p)$, обозначается как $E_p(a, b)$[4].

Типичные варианты графиков эллиптических кривых изображены на рисунке 1.1 [2].

Теорема Хассе утверждает, что, если N – количество точек кривой, определенной над полем Z_p с p элементами, то справедливо неравенство [3]:

$$p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}$$

В ходе работы для заданного p были проанализированы распределения порядков относительно разных величин.

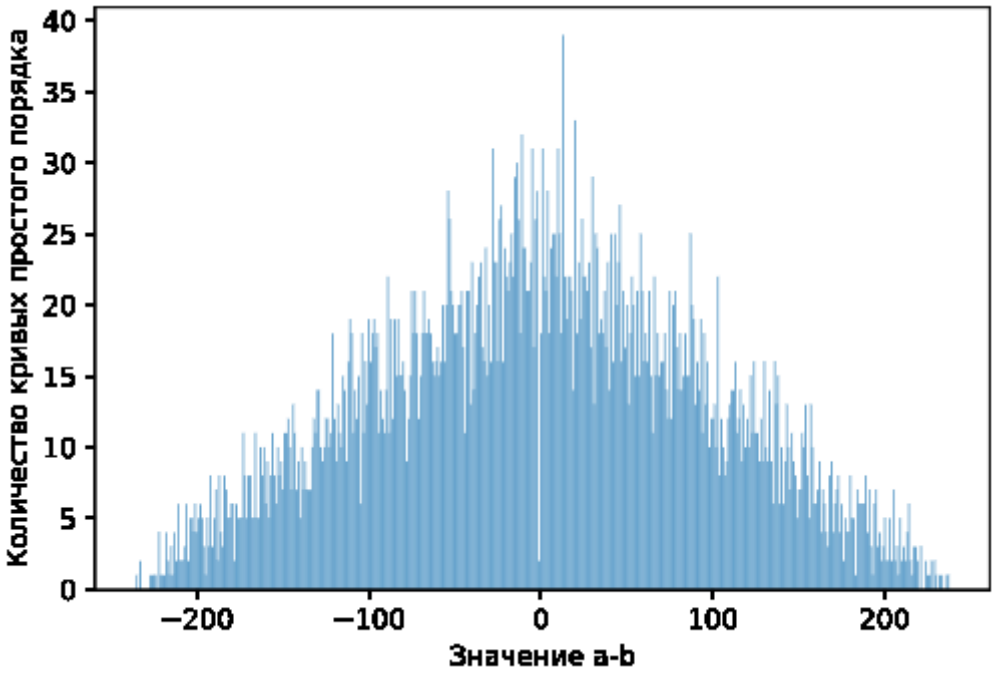


Рис. 1. График зависимости количества кривых простого порядка от выражения a-b

Мы видим, что простые порядки чаще встречаются в середине. Пока что проверим другие функции.

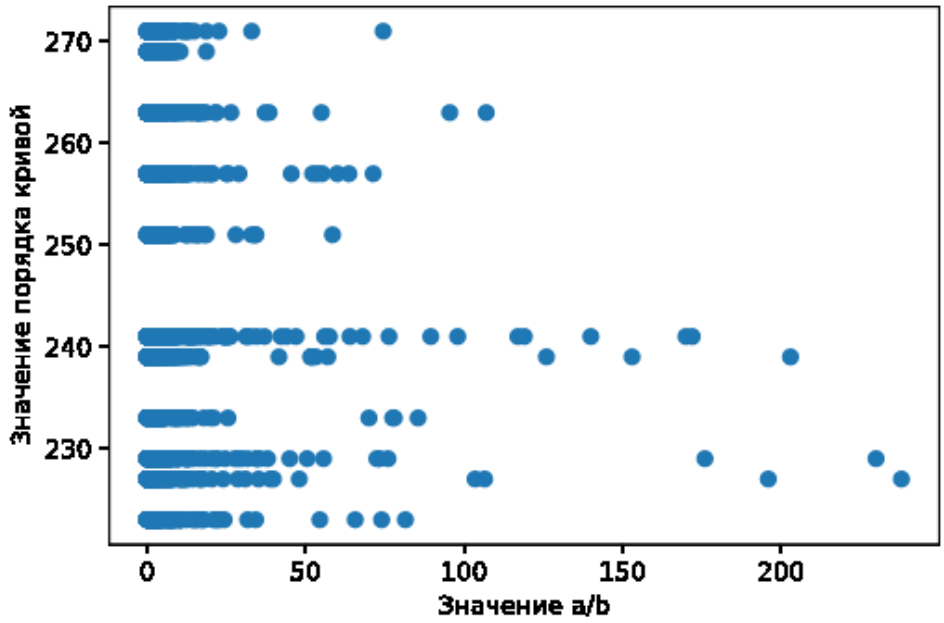


Рис. 2. График зависимости значения простого порядка кривой от выражения a/b.

Из этих рисунков мы можем понять, что простые порядки чаще встречаются при a стремящемся к b. Это важный факт, но для того, чтобы это использовать нужно посмотреть на картину в целом.

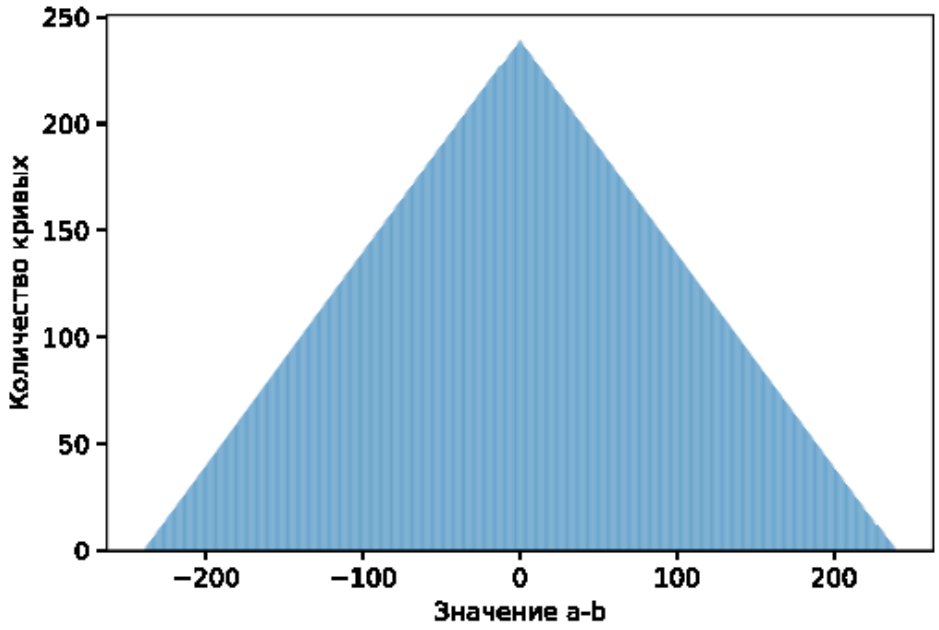


Рис. 3. График зависимости количества кривых от выражения a-b.

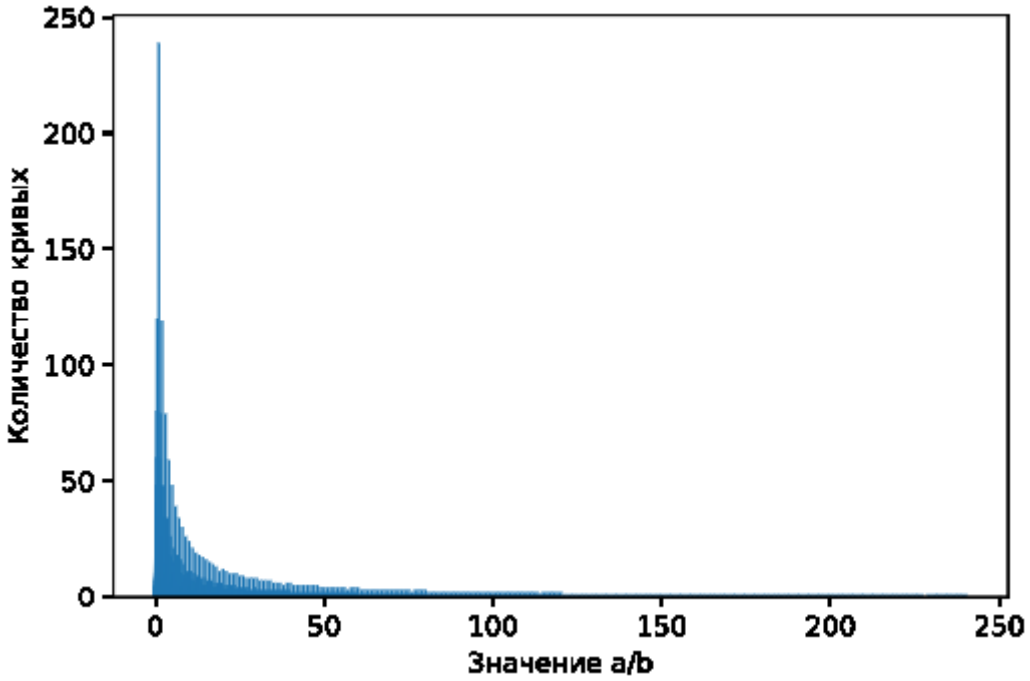


Рис. 4. График зависимости количества кривых от выражения a/b.

Из гистограмм 3, 4 можно понять, что это общая тенденция и найденный факт не получится использовать для генерации кривой.

Библиографический список

1. Об электронной подписи: федеральный закон от 6 апреля 2011 года №63-ФЗ (в ред. от 06.04.2011) // Российская газета. – 2011. – №75.
2. Эллиптическая криптография: теория [Электронный ресурс]. – Режим доступа: http://dml.compkaluga.ru/forum/index.php?showtopic=80815/, свободный. – Загл. с экрана.
3. Жданов, О.Н. Применение эллиптических кривых в криптографии: учебное пособие / О.Н. Жданов, Т.А. Чалкин – Красноярск: Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева, 2008. – 65 с.

STUDY OF STATISTICAL DEPENDENCES OF THE ORDER OF A GROUP OF ELLIPTIC CURVE POINTS ON ITS PARAMETERS

Brovkin I.A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, bibika002@yandex.ru

Weierstrass elliptic curves are considered. General features of algebra in the group of points of elliptic curves are described. Described the use of elliptic curves in cryptography, especially in Russian standards. Methods for the formation of elliptic curves for use in cryptography are investigated. A program has been written that creates a specific selection of elliptic curves. Formed a selection of elliptic curves, for them the order of the group of points is calculated. The data is analyzed using math packages. For this probabilistic model, various hypotheses of dependences are investigated that can affect the probability distribution of a simple order of a group of points of an elliptic curve. It is shown that the distribution of prime orders of a group of points of an elliptic curve does not depend on the parameters of this curve, as well as the differences of these parameters or ratios.

Key words: cryptography, elliptic curves, group theory.

УДК 004.896

НЕЙРОСЕТЕВОЕ МОДЕЛИРОВАНИЕ СЕРДЕЧНО-СОСУДИСТЫХ ЗАБОЛЕВАНИЙ: ПОИСК ЗАКОНОМЕРНОСТЕЙ, ИЗВЛЕЧЕНИЕ ПОЛЕЗНЫХ ЗНАНИЙ

Гилёва Алёна Васильевна, Ясницкий Леонид Нахимович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, Alyona-gil@mail.ru

В статье представлено исследование методом нейросетевого моделирования влияния на состояние сердечно-сосудистой системы общих показателей человека, а также некоторых других факторов, являющимися частью истории жизни пациента. В основе исследования лежит нейро-экспертная система «KardioNet», которая вычисляет степень развития некоторых заболеваний сердца и сосудов. В процессе работы была создана программа для автоматизированного процесса сбора данных с сайта «KardioNet», с ее помощью проведены виртуальные эксперименты для определения влияния следующих показателей пациента: вес, группа крови, время года рождения, курение, физическая нагрузка. Результаты работы могут помочь выявить новые, неизвестные науке зависимости, которые в последующем можно использовать в оптимизации курса лечения сердечно-сосудистых заболеваний.

Ключевые слова: искусственный интеллект, нейросетевые технологии, сердечно-сосудистые заболевания, диагностика заболеваний, прогнозирование.

Заболевания сердечно-сосудистой системы на сегодняшний день являются основной причиной смерти во всем мире. Борьба с ними одна из самых актуальных проблем для врачей, поэтому огромное внимание уделяется совершенствованию диагностики и лечению заболеваний сердца и сосудов.

Использование нейросетей в диагностике заболеваний в различных областях медицины давно используется на практике, причем довольно удачно.

В работе [1] разрабатывается компьютерная нейросетевая модель с целью прогнозирования развития соматической патологии на основе психологических девиаций в личности пациента. В статье [2] рассказывается о разработке моделей прогнозирования прогрессирования хронической болезни почек. Можно выделить серию работ [3-5], в которых используется нейросетевой аппарат прогнозирования временных рядов. Также найдены статьи, посвященные прогнозированию болезни Альцгеймера [6, 7], прогрессирование глаукомы [8], хронической обструктивной болезни легких [9].

Но зачастую эти успехи касаются непосредственно диагностики и классификации заболеваний, хотя применение нейросетевых технологий в медицине позволит по-другому подойти и к проблеме получения новых знаний.

Основная цель настоящей работы заключается в исследовании методом нейросетевого моделирования влияния на состояние сердечно-сосудистой системы общих показателей человека, а также некоторых других факторов, являющимися частью истории жизни пациента.

Пермскими учёными создана нейро-экспертная система «KardioNet», с её помощью человек может узнать вероятность развития у него одного из восьми сердечно-сосудистых заболеваний. Данная система включает в себя три этапа обследования: самостоятельное, первичное и специальное. Первый этап, самостоятельное обследование, даёт предварительные результаты, исходя из общих показателей человека, его истории жизни и жалоб. Именно на этом этапе пациент может прогнозировать развитие обнаруженных заболеваний на будущие годы, изменяя показатель своего возраста, а также посмотреть, как и насколько изменятся эти показатели в случае изменения образа жизни. Последующие этапы требуют более глубоких знаний и требуют помощи специалистов. Самые точные результаты система покажет при заполнении всех трёх этапов обследования.

В работе использован метод «замораживания» [10], суть которого заключается в варьировании значения одного параметра и фиксировании значений остальных. Данный метод позволяет выявить влияние исследуемого параметра на выходной. Но так как человек – это сложный объект моделирования, его входные параметры имеют сложные корреляционные взаимозависимости. Поэтому исследование проводится только для первого этапа обследования системы.

Для изучения влияния входных параметров пациента на результат работы системы были взяты следующие факторы: вес, группа крови, время года рождения, курение, занятие физической нагрузкой.

Для каждого из них проведен отдельный эксперимент, в ходе которого перебраны всевозможные значения данного параметра, при этом каждый раз был изменен один из факторов истории жизни пациента.

При исследовании нам потребовалось обратиться к системе множество раз, меняя различные входные параметры и сохраняя результат запроса. Выполнять эти действия и переносить полученные результаты в Excel самостоятельно – малоэффективно. Было решено автоматизировать данный процесс, что привело к созданию компьютерной программы.

Далее представлены наиболее интересные результаты, которые имеют большую разницу в выходных показателях.

Влияние веса. Рассмотрим поведение показателя вероятности постановки диагноза «Инфаркт миокарда» при увеличении веса взятого нами пациента (см. рис. 1). В какой-то момент цифры резко повышается почти до 27%, но при изменении входных параметров истории жизни пациента эта граница варьируется, также мы можем увидеть, что и при низком весе есть большой риск получить данный диагноз: при сахарном диабете, кардиохирургических вмешательствах или ранее верифицированного заболевания сердечно-сосудистой системы.

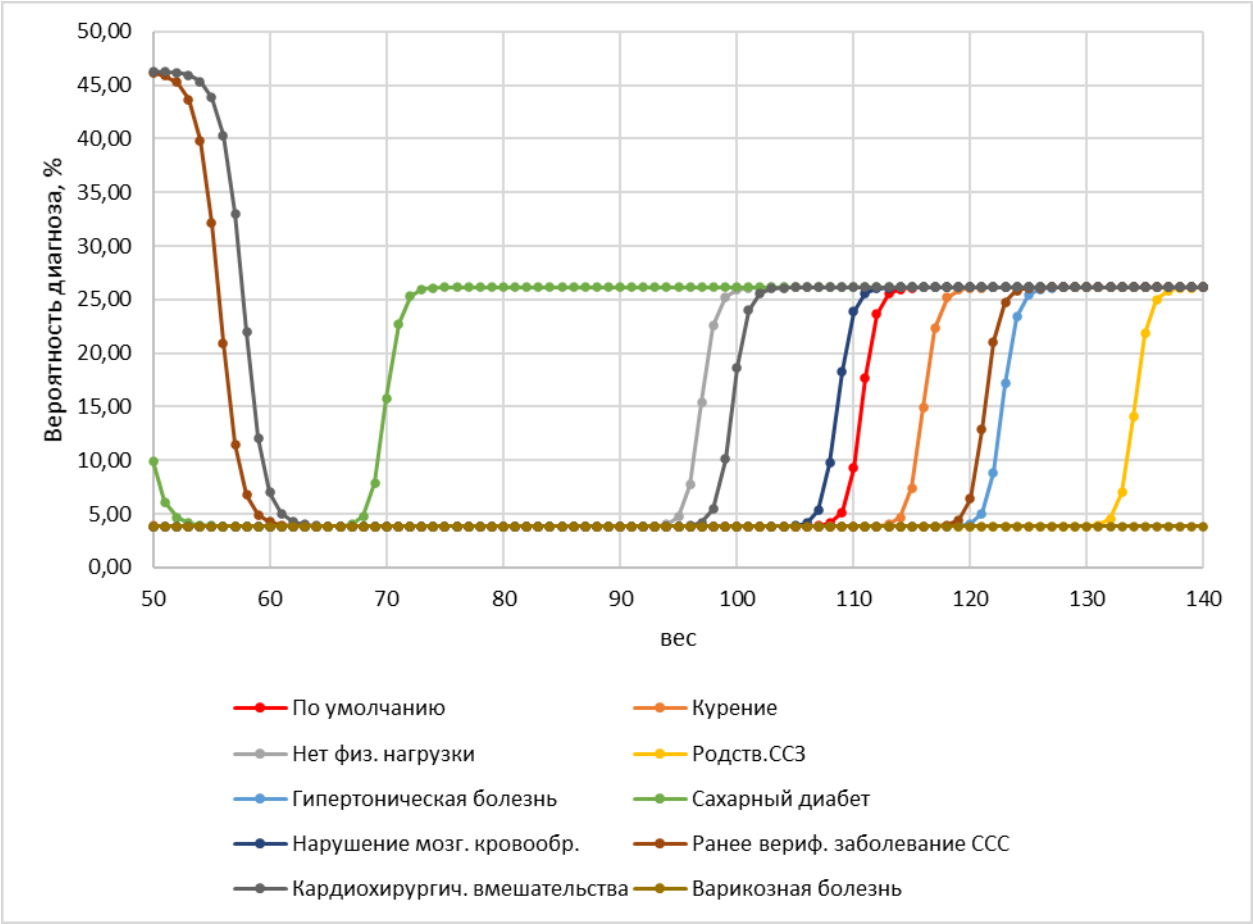


Рис. 1 Вероятность постановки диагноза «Инфаркт миокарда» при изменении веса пациента

Однако некоторые итоги вызвали вопросы с медицинской точки зрения. Для пациента с наличием варикозной болезни или тромбофлебита, вероятность постановки диагноза никак не менялась. Но тромбофлебит и, ведущая к нему варикозная болезнь, ведут к образованию тромбов, а значит впоследствии и к проблемам сердечно-сосудистой системы.

Также мы видим, что порог для пациента, у кровных родственников которого есть заболевания сердечно-сосудистой системы, по сравнению с абсолютно здоровым человеком, «сдвигается вправо» почти на 10 кг, хотя данный показатель считается фактором риска для пациента. То же самое касается и других показателей.

Влияние группы крови. Врачи не выделяют группу крови в факторы риска заболеваний сердечно-сосудистой системы, поэтому мы не можем при анализе результатов сравнить наши данные с рекомендациями врачей, это совершенно новые знания.

Наиболее существенные различия в выходных данных наблюдаются в постановке диагноза «Инфаркт миокарда» (см. рис. 2). Например:

1) Пациент с ранее верифицированным заболеванием сердечно-сосудистой системы. В зоне риска находятся люди с четвёртой положительной группой крови. Вероятность получить данный диагноз для рассматриваемого нами пациента при всех тестах не превышал 7%, но при изменении группы крови на IV+ этот показатель вырос до 46%.
2) Пациент, перенесший кардиохирургические вмешательства. Здесь мы видим, что 2 значения выбиваются из общего графика: пациенты с третьей и четвёртой положительной группой крови. Здесь столбики с показателями устремляются вверх и достигают 22% и 46% соответственно.

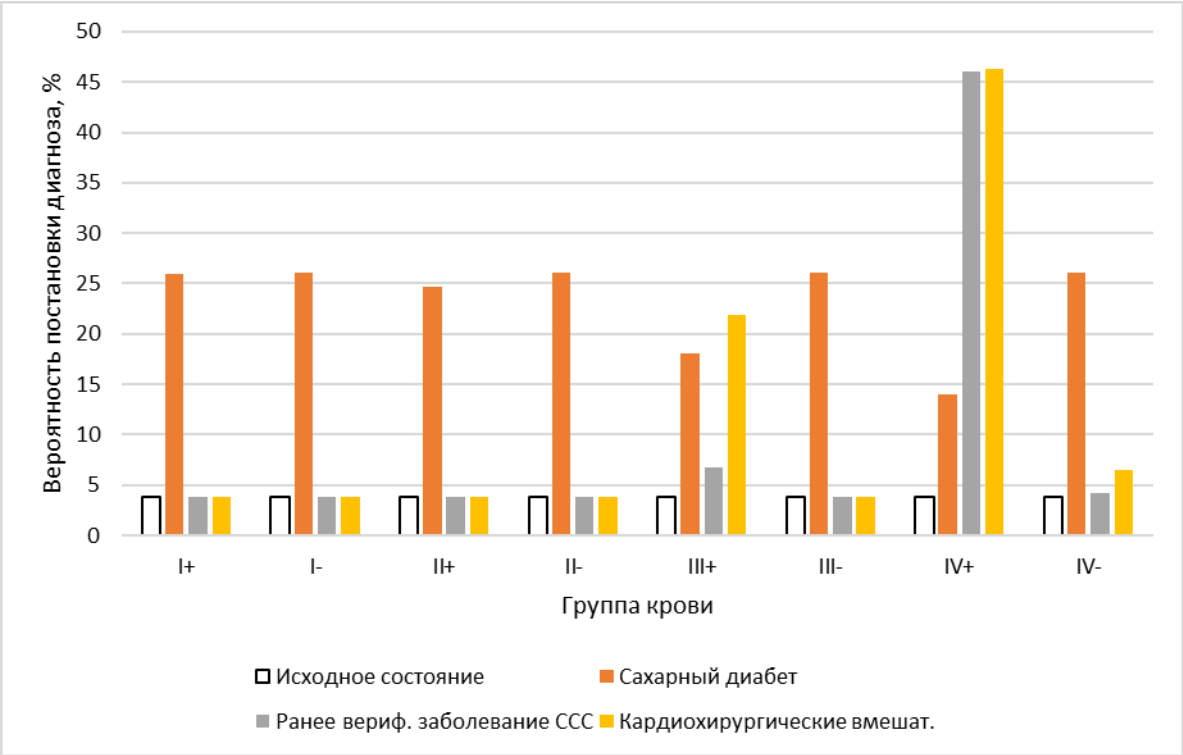


Рис. 2 Вероятность постановки диагноза «Инфаркт миокарда» при изменении группы крови пациента

Влияние времени года рождения. Время года рождения пациента, как и в случае с группой крови, не рассматривается врачами как фактор риска.

Результаты показали, что для взятого нами пациента, дата рождения которого приходится на летний период, вероятности постановки диагноза «Стенокардия стабильная» наиболее высоки по сравнению с остальными временами года (см. рис. 3). Особенно большие перепады наблюдаются для пациентов с сахарным диабетом или ранее верифицированным заболеванием сердца и сосудов, а также для тех, кто курит, не занимается физическими нагрузками.

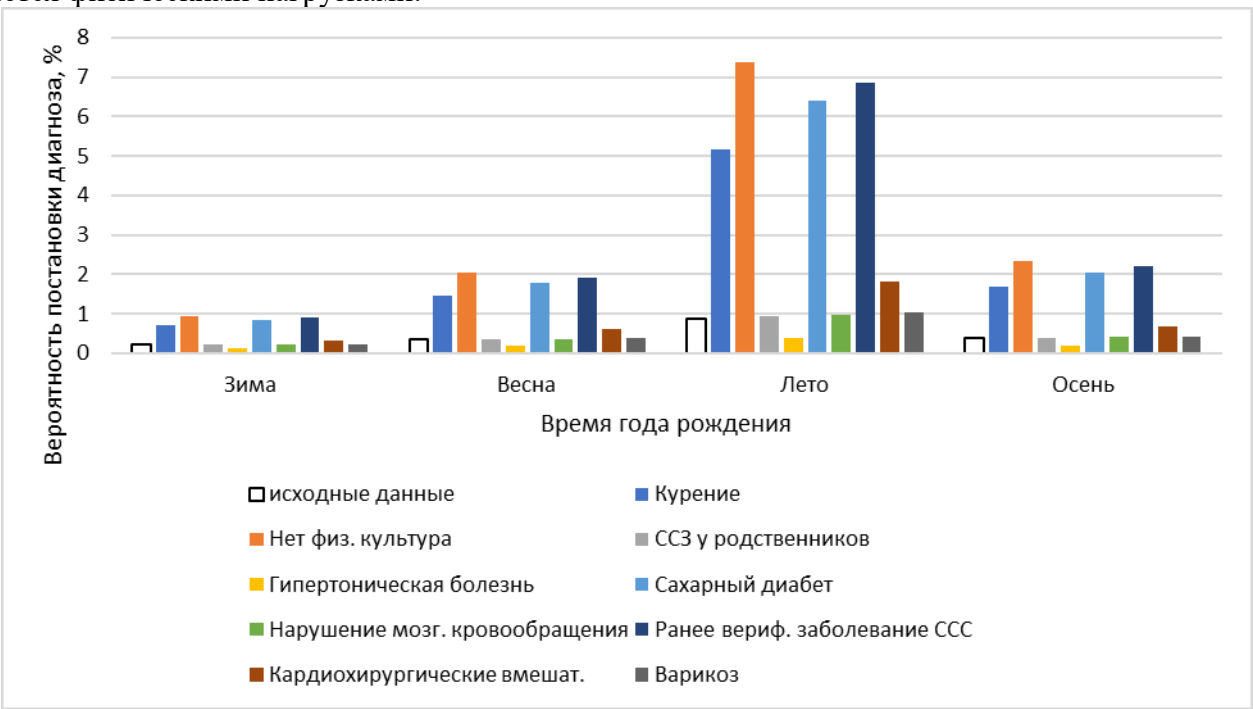


Рис. 3 Вероятность постановки диагноза «Стенокардия стабильная» при изменении времени года рождения пациента

При изменении параметров истории жизни пациента в целом вероятность получить диагноз «Инфаркт миокарда» оставалась на одном уровне, но исключением стал пациент с наличием сахарного диабета, который родился весной, для него данный показатель вырос до 22,45%, при том, что для таких же пациентов, дата рождения которых приходится на другие времена года, вероятность равна 3,82%.

Влияние курения. Итоги данного эксперимента показали, что курение не всегда негативно влияет на здоровье человека. Для пациента с сахарным диабетом вероятность постановки диагноза «Инфаркт миокарда» при курении снижается более, чем на 20% (см. рис. 4). В остальном же курение негативно сказывается на показателях остальных диагнозов.

Здесь же мы снова обратились к врачу, и данный результат оказался противоречивым: курение относится к одним из главных факторов риска атеросклероза, что напрямую влияет на развитие различных сердечно-сосудистых заболеваний.

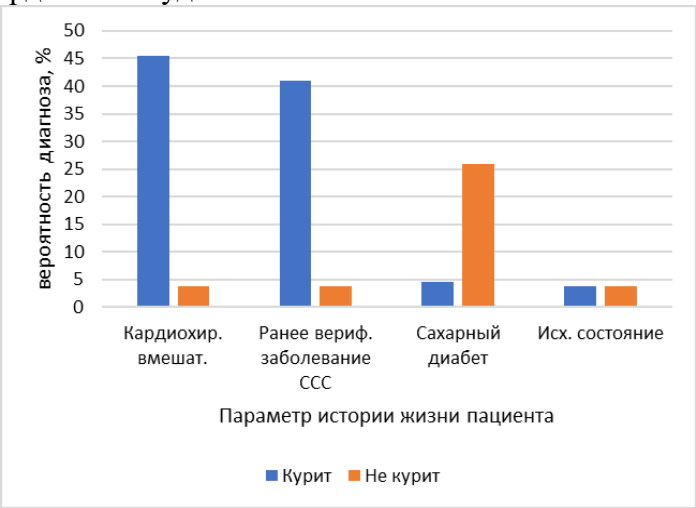


Рис. 4 Вероятность постановки диагноза «Инфаркт миокарда» для курящего и некурящего пациента

Влияние физической нагрузки. В данном исследовании мы не получили ярко выраженных результатов. В целом занятие легким видом спорта или физической зарядкой понижают вероятности постановки представленных диагнозов, однако разница в этих различиях невелика.

Заключение. В нашей работе мы рассмотрели возможность прогнозировать развитие заболеваний сердца и сосудов с помощью нейросетевой медицинской системы.

Анализируя полученные результаты, мы видим, что чаще всего рекомендации врачей справедливы: физическая нагрузка, курение, увеличение веса человека негативно влияют на здоровье пациента. Однако система «KardioNet» показала и неожиданные результаты, которые оспаривают общепризнанные факторы риска и не имеют медицинского обоснования.

Также нами были проведены исследования в поиске абсолютно новых, неизвестных зависимостей, которых нет в медицинских источниках: влияние на постановку диагноза времени года рождения и группа крови пациента. Мы получили интересные выводы. Особенно интересно влияние группы крови пациента на постановку диагноза «Инфаркт миокарда», разница между выходными параметрами системы достигала более 40%. Результаты при проверке влияния времени года рождения человека лежат в пределах погрешности системы, поэтому нельзя утверждать, что данные зависимости имеют место быть в жизни.

Возможно, для получения более точных и развернутых результатов необходимо провести ряд тестов с другими пациентами, не только меняя какой-либо один показатель истории жизни пациента, но и комбинируя эти параметры между собой, что можно исследовать в дальнейшем с помощью разработанной программы для автоматизации запросов к системе.

Библиографический список

1. Прохореко И. О. Метод нейросетевого моделирования и его использование для прогнозирования развития соматической патологии у лиц старших возрастных групп // Современные проблемы науки и образования. 2013. № 1. [Электронный ресурс] URL: https://www.science-education.ru/ru/article/view?id=8411 (дата обращения 18.04.2021).
2. Tangri N, Stevens L, Griffith J, et al. A predictive model for progression of chronic kidney disease to kidney failure. Journal of the American Medical Association, 2011, vol. 305, iss. 15, pp. 1553–1559.
3. Gan R., Chen X., Yan Y., Huang D. Application of a hybrid method combining grey model and back propagation artificial neural networks to forecast hepatitis b in china. Computational and Mathematical Methods in Medicine, 2015, 7 p.
4. Zecchin C., Facchinetti A., Sparacino G., Cobelli C. Jump neural network for real-time prediction of glucose concentration. Methods in Molecular Biology, 2015, vol. 1260, pp. 245–259.
5. De Schatz C.H.V., Schneider F.K., Abatti P.J., Nievola J.C. Dynamic Fuzzy-Neural based tool formonitoring and predicting patients conditions using selected vital signs. Journal of Intelligent and Fuzzy Systems, 2015, vol. 28, iss. 6, pp. 2579–2590.
6. Sukkar R, Katz E, Zhang Y, Raunig D, Wyman B. Disease progression modeling using hidden Markov models. Engineering in Medicine and Biology Society, 2012, pp. 2845–2848.
7. Zhou J, Liu J, Narayan V, Ye J. Modeling disease progression via multi-task learning, 2013, pp. 233–248.
8. Liu Y-Y, Ishikawa H, Chen M, Wollstein G, Schuman J, Rehg J. Longitudinal modeling of glaucoma progression using 2-dimensional continuous-time hidden Markov model. Medical Image Computing and Computer-Assisted Intervention, 2013 pp. 444–451.
9. Wang X, Sontag D, Wang F. Unsupervised learning of disease progression models. Knowledge Discovery and Data Mining, 2014, pp. 85–94.
10. Ясницкий Л.Н. Введение в искусственный интеллект. М.: Издательский центр «Академия», 2005. – 176 с.

NEURAL NETWORK MODELING OF CARDIOVASCULAR DISEASES: SEARCH FOR REGULARITIES, EXTRACTION OF USEFUL KNOWLEDGE

Gilyova Alyona V., Yasnitsky Leonid N.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, yasn@psu.ru

The article presents a study of the effect on the state of the cardiovascular system of general indicators of a person and some other factors that are part of the patient's life history by the method of neural network modeling. The research is based on the neuro-expert system "KardioNet", which calculates the degree of development of certain diseases of the heart and blood vessels. In the process of work, a program was created for the automated process of collecting data from the site "KardioNet", with its help, virtual experiments were carried out to determine the influence of the following patient indicators: weight, blood type, time of birth, smoking, physical activity. The results of the work can help to identify new addictions unknown to science, which can later be used to optimize the course of treatment of cardiovascular diseases.

Keywords: artificial intelligence, neural network technologies, cardiovascular diseases, diagnosis of diseases, forecasting.

УДК 004.942

ВЫБОР ИНСТРУМЕНТА ПРОГНОЗИРОВАНИЯ ПОТРЕБЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ

Зарубин Дмитрий Сергеевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, dimaster22@gmail.com

Целью данной работы является сравнение готовых инструментов для прогнозирования временных рядов, представляющих метрики основных вычислительных ресурсов. Прогнозирование потребности вычислительных ресурсов может оказаться полезным при решении закупки оборудования, оптимизации работы запущенных приложений или выборе инструментов или образов для развёртки приложений на серверах или в контейнерах. С другой стороны, прогнозирование не может помочь в обнаружении критических ситуаций, не связанных с работой прогнозируемой системы, но влияющими на неё. В статье представлены сравнение двух инструментов и их работы на временном ряде загруженности ОЗУ.

Ключевые слова: временной ряд, прогнозирование, вычислительные ресурсы.

Введение

С наращиванием производственных масштабов, в компании возникает необходимость в контроле больших мощностей. Каждый сервис на серверах компании запускается в контейнере – оболочке для приложения, в которую также упакованы все зависимости такие, как библиотеки, настройки, инструменты и среда запуска. Контейнеры соединяются кластер, дублируются, чтобы предотвратить сбои или работать на разных машинах. Для мониторинга ресурсов, потребляемыми контейнерами, устанавливаются различные системы мониторинга. Данные, полученные системой мониторинга, попадают к специалисту, который принимает решения о перераспределении ресурсов, обновлении аппаратной или программной части. Эти решения зависят только от компетенции или интуиции специалиста. Чтобы увеличить шансы на верное решение, предлагается прогнозировать динамику ресурсов. Необходимо делать это в реальном времени и на любых данных, без долгого обучения работы с инструментами или для простоты встраивания для автоматического прогнозирования, поэтому необходимо подобрать общее решение.

PMD ARIMA

ARIMA (AutoRegressive Integrated Moving Avarage) – модель для прогнозирования временных рядов. Модель является развитием модели ARMA[1], она использует дополнительный компонент I, который нужен для избавления временного ряда от трендовой составляющей, путём интегрирования. Количество степеней интегрирования выражено в параметре d. Определение параметров ARIMA определяется по коррелограмме и статистическим тестам. Модель ARIMA нужна для того, чтобы предсказывать данные с трендом, но не может справиться с данными с ярко выраженными сезонными компонентами. Для решения этой задачи, существует модель SARIMA[2], добавляющая возможность работы с сезонностью.

PMD ARIMA – это библиотека на языке Python, которая содержит функцию autoarima, которая создаёт модели ARMA, ARIMA и SARIMA с различными параметрами и выбирает лучшую по критерию Акаике[3]. Это позволяет не исследовать временной ряд вручную в поисках вида модели и подбора коэффициентов для построения модели.

Качество обучения часто зависит от данных и количества дополнительных сезонных описаний. Модель плохо реагирует на пропуски в данных, что не подходит для предсказания ресурсов. Пропуски в данных могут быть вызваны сбоями, перезагрузками или отключениями сервера, которые могут случаться довольно часто.

Facebook Prophet

Facebook Prophet – библиотека для прогнозирования временных рядов от компании Facebook. Она использует модель AM[4] для предсказания данных с сильно выраженной сезонностью.

Инструмент позволяет гибко настроить модель, выбрав преобразование данных и заранее известные аномальные дни. Не нужно подбирать модель, что значительно экономит время. Также инструмент устойчив к пропускам в данных, сдвигам тренда. Но плохо справляется с прогнозами данных с маловыраженной сезонностью. Для построения более качественной модели создатели рекомендуют, как минимум, данные за год для обучения.

Сравнение работы инструментов

Для сравнения качества прогнозов была выполнено обучением модели, полученной с помощью PMD ARIMA и стандартного использования FB Prophet. PMD ARIMA потребовала первоначального преобразования данных, исключения выбросов и пустых мест, после чего при помощи функции autoarima была предложения модель с параметрами D=1, d=1, Q=1, q=3, P=1, p=4. Подбор параметров занял два часа для тестовой выборки в год. FB Prophet справился с прогнозом значительно быстрее и не потребовал дополнительных преобразований. Для оценки качества прогнозов были использованы меры RMSE и MSE. Результаты представлены в таблице 1.

Таблица. 1 Результаты ошибок прогнозов

Инструмент	RMSE	MSE
PMD ARIMA	9.019815	60.870360
FB PROPHET	12.642057	126.711041

Заключение

Были сравнены библиотеки PMD ARIMA и Facebook PROPHET. Прогноз выполненный с помощью PMD ARIMA показал более точный результат, но его достижение требовало больше времени, также метод не подходит для прогнозирования данных в реальном времени и автоматическом режиме из-за необходимости предобработки данных для обучения.

Библиографический список

1. *ARMA* Processes. University of California, Davis; 2021. [Электронный ресурс] URL: https://stats.libretexts.org/@go/page/812 (дата обращения: 21.04.2021)
2. *A Gentle* Introduction to SARIMA for Time Series Forecasting in Python [Электронный ресурс] URL: https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python (дата обращения: 21.04.2021)
3. *Akaike's* information criterion: Definition, Formulas [Электронный ресурс] URL: https://www.statisticshowto.com/akaikes-information-criterion (дата обращения: 21.04.2021)
4. *Charles J. Stone* "Additive Regression and Other Nonparametric Models,"// Книга: The Annals of Statistics, Ann. Statist. 1985, с.689-705

CHOOSING FORECASTING INSTRUMENT FOR COMPUTING RESOURCES TIME SERIES

Zarubin Dmitry S.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, dimaster22@gmail.com

The purpose of this paper is to compare ready-made tools for forecasting time series of basic computing resources. Computing resources forecasting can be useful to decide such problems as purchasing hardware, optimizing running applications or choosing ways to deploying applications to servers or containers. On the other hand, forecasting can't help to detect critical situations not related to the operation of the predicted system, but affecting it. The article presents a comparison of two tools and their work on the time series of the used RAM.

Keywords: time series, forecasting, computing resources.

УДК 003.26.09

ИССЛЕДОВАНИЕ АТАКИ ПО ВРЕМЕНИ НА БЛОЧНЫЙ ШИФР ГОСТ 34.12-2018 «КУЗНЕЧИК»

Зимников Дмитрий Александрович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, zdpkfh@mail.ru

В данной работе исследуется перспективность атаки по времени на блочный шифр ГОСТ 34.12-2018 с длиной ключа шифрования 256 бит и длиной блока 128 бит, также известный как «Кузнечик». Проводится краткий теоретический обзор блочного шифра «Кузнечик», создание программы, генерация данных и исследование зависимости времени выполнения от входных данных и сравнение выходных данных для двух различных случаев: с использованием операторов ветвления и с использованием битовых масок. Рассматривается возможность или невозможность атаки по времени на шифр и методы противодействия атаке. Полученные результаты исследования анализируются, на их основе принимается решение о наличии или отсутствии возможности атаки на блочный шифр «Кузнечик», создаются рекомендации для противодействия подобным атакам.

Ключевые слова: Кузнечик, атака по времени, программная атака по времени, криптоанализ, атака на блочный шифр.

В наше время огромную роль играют электронные средства передачи, хранения, и обработки информации, в том числе компьютеры. Ввиду постоянного использования информационных технологий в различных областях деятельности и в повседневной жизни, необходимо обеспечивать их безопасность.

Для предотвращении несанкционированного доступа к данным (обеспечению конфиденциальности данных) – используется шифрование данных, с целью защиты информации в Российской Федерации введен в действие межгосударственный стандарт ГОСТ 34.12-2018 в качестве национального стандарта Российской Федерации с 1 июня 2019 г.

Таким образом стойкость блочного шифра «Кузнечик» имеет большое значение для безопасности информации в Российской Федерации. Одним из способов анализа алгоритмов шифрования, позволяющих получать секретную информацию при зашифровании или расшифровании, несмотря на криптографическую стойкость алгоритма, являются атаки по побочным каналам, осуществляемые путём многократного наблюдения какой-либо «утечки» информации – информации о потребляемой электроэнергии, электромагнитном излучении, времени исполнения и другим физическим параметрам.

Большая часть атак по побочным каналам предполагает неоднократное наблюдение некоторой информации о работе реализации алгоритма, не являющейся её непосредственным входом или выходом, после чего в условиях заранее выбранной вероятностной модели наблюдений выдвигаются и проверяются статистические гипотезы о фактических значения ключевой информации.

В случае с блочными шифрами, к которым и относится «Кузнечик», наиболее часто наблюдение происходит за следующими типами информации:

- Ключевая информация – содержит в себе значение непосредственно используемого секретного ключа шифрования.
- Выход определённого такта работы блочного шифра. Из такой информации, при известных открытых текстах, можно вычислить значения секретного ключа.

Основными причинами, позволяющими провести атаку по времени, являются оптимизации производительности, команды процессора, исполняющиеся за недетерминированное время – такие как умножение и сложение, операторы ветвления.

Одним из способов защиты от атаки по времени является исключение из реализации алгоритмов операторов ветвления, другим – разработка аппаратных платформ, которые исполняют операции за детерминированное время, третьим – доведение времени работы до заранее заданного значения, превышающего время выполнения любой из требующих защиты операций.

Для блочного шифра DES при помощи атаки по времени возможно восстановление последнего раундового ключа при 100000 исходных текстов с замерами времени исполнения. После его получения, учитывая алгоритм формирования раундового ключа, сложность перебора ключа уменьшается с 2^{56} до 2^{8} [1].

Относительно работы шифра «Кузнечик» можно заметить, что что итерационные ключи создаются на основе изначального секретного ключа при помощи обратимых преобразований, поэтому для восстановления исходного секретного ключа достаточно восстановления последних двух парных раундовых ключей (K_{2i+1}, K_{2i+2}) : используя обратные преобразования будет восстановлена пара раундовых ключей $(K_1, K_2), K = K_1 \| K_2$ [2].

Для того, чтобы получить итерационные ключи можно попробовать воспроизвести известные методы атаки на блочные шифры. Для этого необходимо обратить внимание структуру шифра: линейное преобразование $\ell : V_8^{16} \to V_8$, использует умножение и сложение в поле Галуа. В свою очередь линейное преобразование используется в преобразовании R, а преобразование R – в преобразовании L. То же справедливо и для обратных преобразований. В большинстве реализаций умножение и сложение в поле Галуа использует операторы ветвления.

Обозначим время работы операции как τ , время работы функции как T , вхождение в оператор ветвления как Δ . Тогда для рассматриваемой функции умножения в поле Галуа время работы алгоритма можно будет записать следующим образом: $T = 3\tau + \Delta\tau + \Delta\tau$, где Δ может принимать значения равные 0 и 1. Очевидно, что в зависимости от значения Δ будет изменяться и время работы функции, а следовательно, и всей программы.

Было реализовано два варианта функции: с использованием оператора ветвления и с использованием битовых масок.

В ходе работы было сгенерировано 200 000 исходных текстов и 8 ключей. Входные данные для программы выглядели следующим образом: файл с ключами шифрования содержал 4 ключа, файл с исходными текстами содержал 100 000 текстов. Для уменьшения количества шума в данных шифрование текста ключом шифрования осуществлялось 1000 раз для каждого текста.

В данной работе исследуется зависимость t от H_1, H_2, H_3, H_4. При этом замечено, что для ключа, которым было осуществлено шифрование, график распределения времени исполнения в зависимости от H заметно отличается от графиков распределения времени исполнения других ключей.

На рисунках 1 и 2 представлены графики при использовании для шифрования текстов ключей №1 – на диаграмме обозначены синим цветом, ключи №2, 3 и 4 обозначены красным, жёлтым и зелёным соответственно. Входные данные для алгоритма: 4 различных ключа длинной 256 бит, 100 000 различных исходных текстов, для каждого текста осуществляется 1000 с использованием выбранного ключа шифрования. При равенстве H для различных исходных текстов берётся их усреднённое значение.

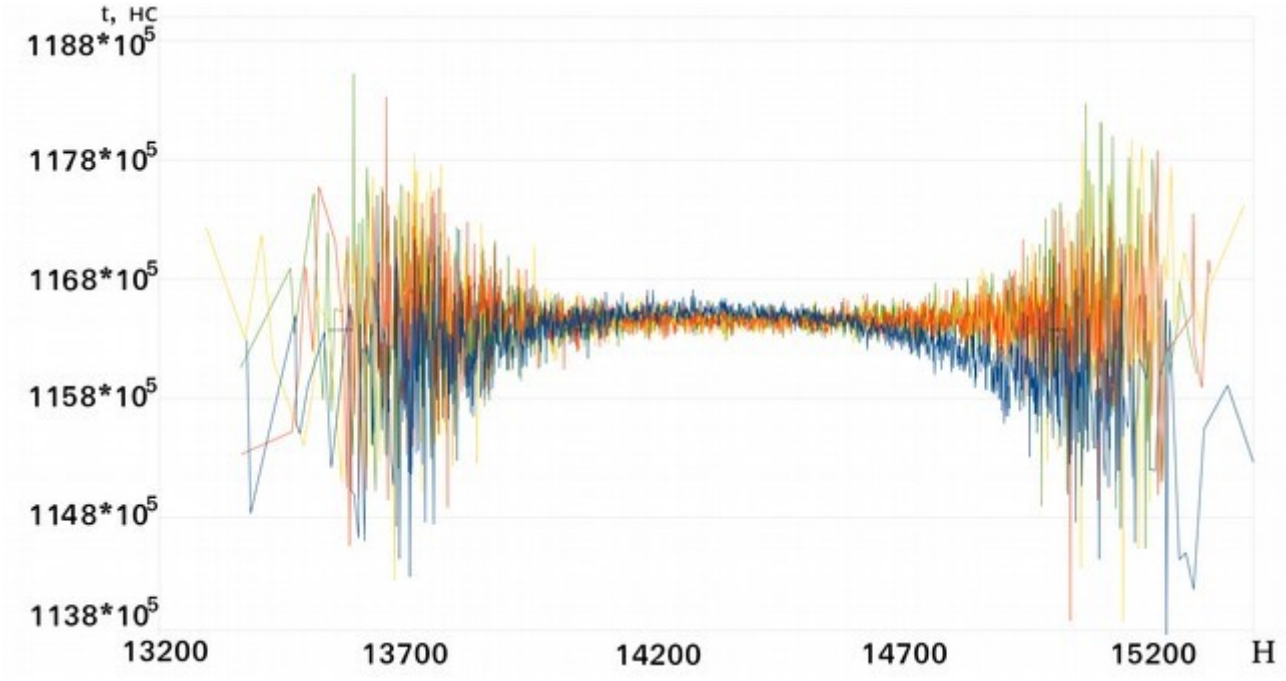


Рис. 1. График зависимости t от H, графики всех ключей в одной системе отсчёта, наложены друг на друга, замер №1, недетерминированное время выполнения.

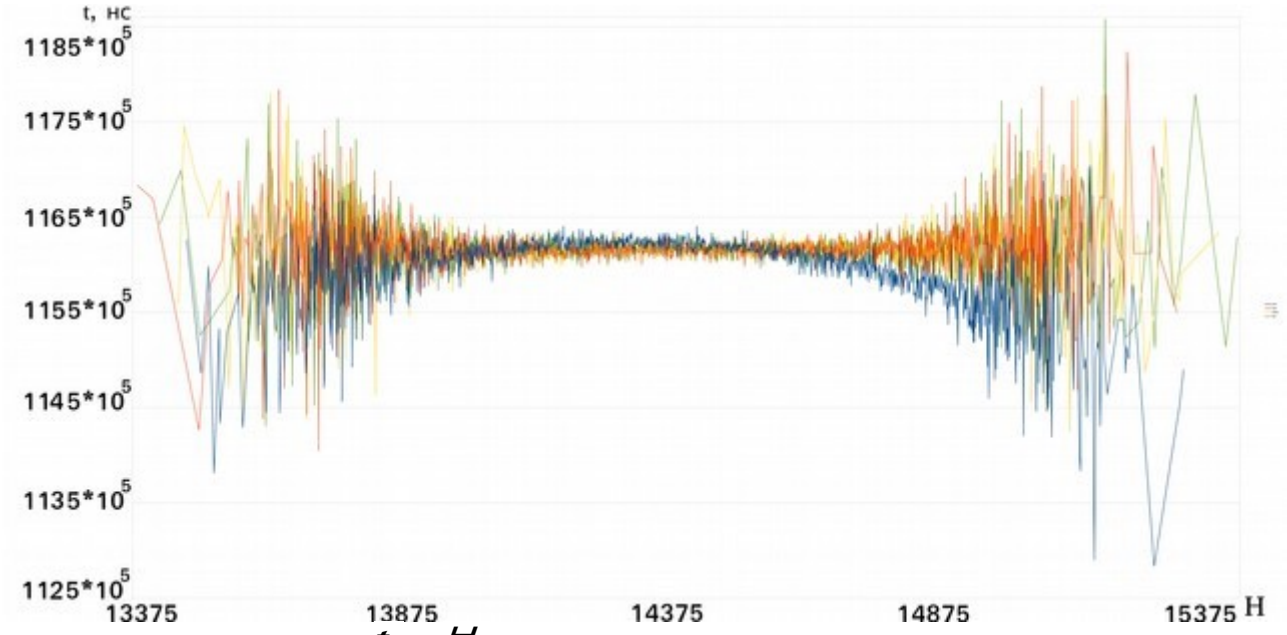


Рис. 2. График зависимости t от H, графики всех ключей в одной системе отсчёта, наложены друг на друга, замер №2, недетерминированное время выполнения.

Как можно заметить, график ключа №1 визуально отличается от других и может быть приблизительно описан как квадратичная функция, в то время как графики других ключей по мере приближения к центральной области выборки могут характеризоваться линейной функцией, что может позволить составить статистическую модель.

При использовании модифицированной версии операций над полями Галуа, основанной на побитовых операциях и битовых масках без операторов ветвления, графики зависимости t от H для тех же входных данных выглядят следующим образом:

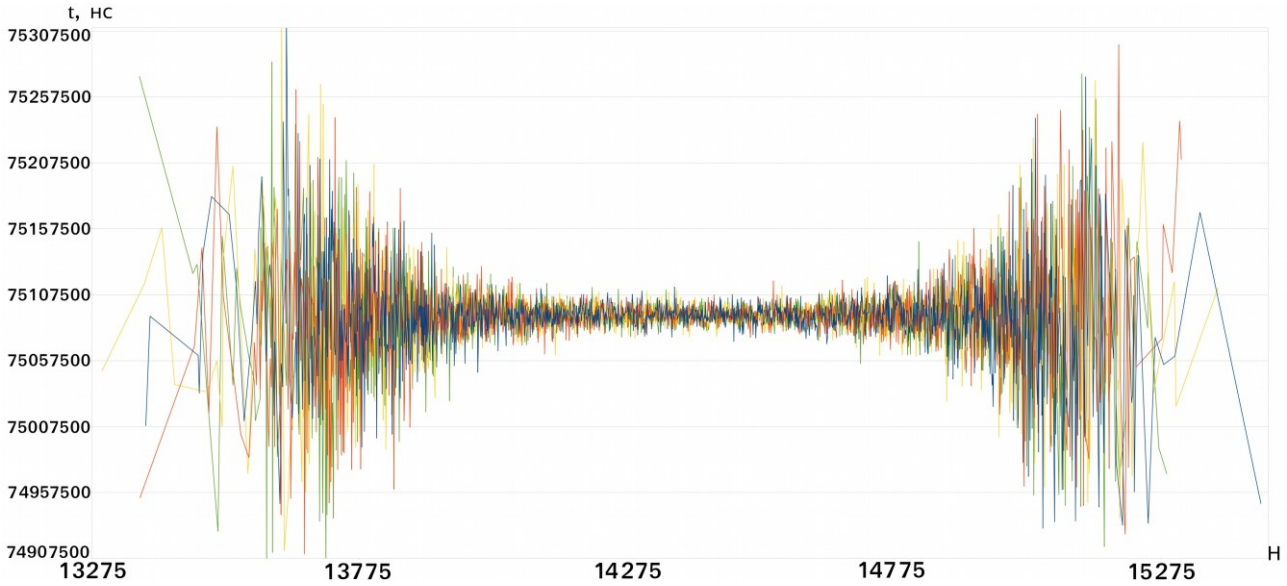


Рис. 3. График зависимости t от H, графики всех ключей в одной системе отсчёта, наложены друг на друга, замер №1, детерминированное время выполнения.

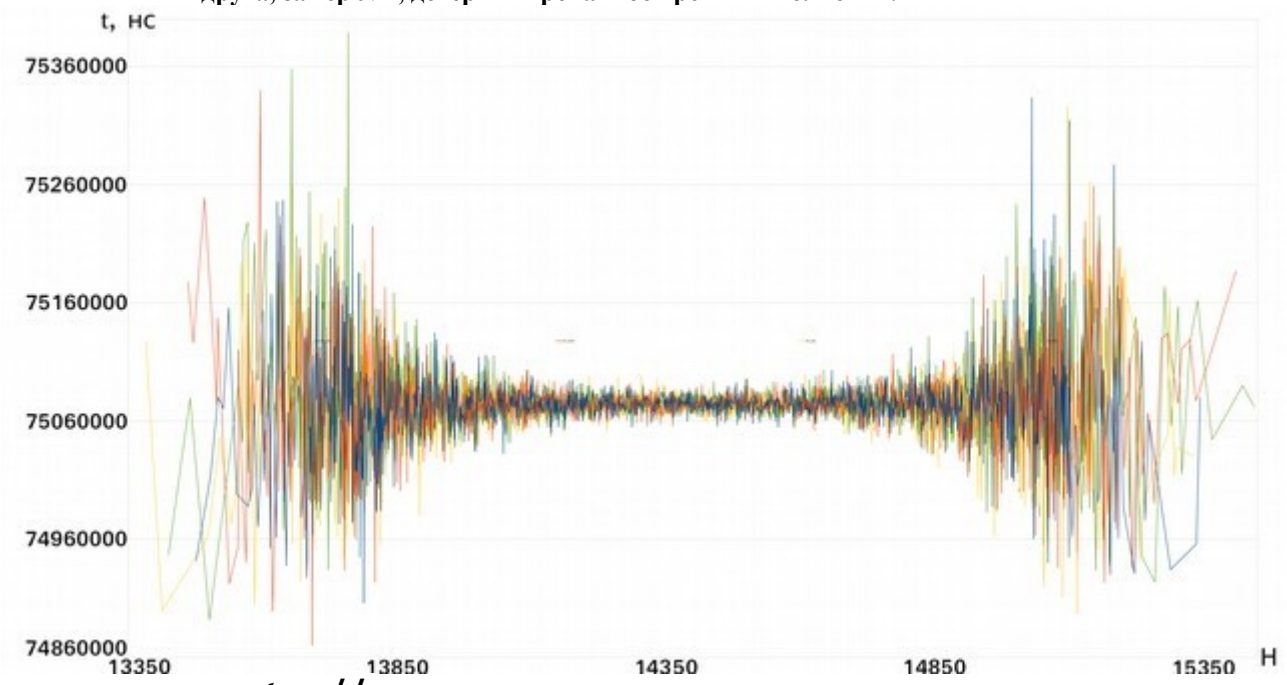


Рис. 4. График зависимости t от H, графики всех ключей в одной системе отсчёта, наложены друг на друга, замер №2, детерминированное время выполнения.

Как можно увидеть на рисунках 3 и 4, при использовании версии алгоритма с детерминированным временем исполнения график зависимости t от H ключа шифрования не имеет явного статистического и визуального отличия от остальных ключей и для создания статистической модели нет предпосылок ввиду статистической неразличимости данных, что означает, что вторая реализация алгоритма, использующая битовые маски, успешно исключает возможность подобной атаки. Вероятность же получения злоумышленником данных, необходимых для построения статистических моделей или же графиков, подобных 1 и 2 , крайне мала ввиду размеров итерационных ключей, что делает задачу сложновычислимой, из чего можно сделать вывод, что атака по времени на шифр «Кузнечик» не является успешной.

Библиографический список

1. DES timing attack *https://github.com/ChristianPalmiero/DES-Timing-Attack*
2. V. A. Kiryukhin. Related-key attack on 5-round Kuznyechik. МАТЕМАТИЧЕСКИЕ ВОПРОСЫ КРИПТОГРАФИИ 2020 Т. 11 No 2 С. 53–67

ANALYSIS OF TIMING ATTACK ON BLOCK CIPHER GOST 34.12-2018 «KUZNYECHIK»

Zimnikov Dmitriy A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, zpdkfh@mail.ru

Abstract. The goal of this paper is to examine the feasibility of timing attack on Kuznyechik GOST cipher with key length 256 bit and block lenght 128 bit. A brief theoretical overview of the "Grasshopper" block cipher is carried out. We created the implementation of it, generated data and searched for relation between time and input data for two different implementations: one where branching was used and another one where bit masks were used instead. Based on results of the research, we decide whether timing attack is feasible and make recommendations to counter such attacks.

Keywords: Kuznyechik, GOST-Grasshopper, timing attack, software timing attacks cryptanalysis, block cipher.

УДК 004.031.43

ВЫБОР БРОКЕРА СООБЩЕНИЙ ДЛЯ ИСПОЛЬЗОВАНИЯ В КАЧЕСТВЕ ШИНЫ ДАННЫХ В РАМКАХ ТЕЛЕКОММУНИКАЦИОННОЙ КОМПАНИИ

Калугин Василий Алексеевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, vasyakalugin@mail.ru

В рамках данной работы проводится сравнение брокеров данных для использования в качестве шины данных и дальнейшего перехода от монолитной архитектуры информационной системы к архитектуре на основе микросервисов в телекоммуникационной компании. Рассмотрены пути увеличения производительности информационной системы путём горизонтального и вертикального масштабирования. Выбран вариант горизонтального масштабирования, как более универсальный и доступный. Проведено сравнение брокеров данных Apache Kafka, RabbitMQ и NATS. Выявлены достоинства и недостатки каждого продукта, рассмотрены технические характеристики и особенности эксплуатации. Проведено попарное сравнение Apache Kafka и RabbitMQ, Apache Kafka и NATS. В результате анализа выявлено, что наиболее оптимальном продуктом является Apache Kafka, поскольку RabbitMQ и NATS не удовлетворяют требованиям по надёжности передачи данных в рамках телекоммуникационной компании.

Ключевые слова: брокер данных, монолитная архитектура, микросервисная архитектура, горизонтальное масштабирование, вертикальное масштабирование, Apache Kafka, RabbitMQ, NATS

Ведение

Современные телекоммуникационные компании часто сталкиваются с необходимостью передачи больших объёмов данных за единицу времени: например, информация о телефонных звонках, данные с различного клиентского оборудования. Сами по себе такие сообщения не очень большие, однако их количество и частота передачи

требует высокой пропускной способности не только от сети, по которой они передаются, но и от базы данных, в которую они помещаются.

Ещё больше усложняет ситуацию то, что часто полученные данные необходимо не только хранить, но и использовать в других сервисах компании – например, выводить клиенту информацию об остатке трафика на его тарифе или для мониторинга стабильности работы телевидения. Такая необходимость в предоставлении информации сразу множеству сервисов-потребителей приводит к значительной нагрузке всей базы данных и, как следствие, к возможным перебоям в доступности данных.

К основным способам увеличения производительности информационной системы можно отнести: горизонтальное или вертикальное масштабирование, переход на микросервисы или же внедрение шины данных. В свою очередь шину данных можно назвать переходным этапом к микросервисной архитектурой, когда каждый сервис работает только со своими данными и не нагружает другие системы, от монолитной архитектуры – когда данные хранятся в одной базе данных и все сервисы работают непосредственно с ней.

В качестве шин данных в высоконагруженных системах можно использовать так называемые брокеры данных. Данное программное обеспечение не нагружено бизнес-логикой, различными средствами для аналитики, а служит только для приёма и маршрутизации большого количества данных.

Брокеры данных взяли на себя такие задачи как преобразование сообщений, надежная транспортировка, длительная оркестровка между конечными потребителями и многое другое. Однако все процесса преобразования и обработки данных ложатся на конечных потребителей (микросервисы).

Самыми популярными брокерами данных можно назвать Apache Kafka, RabbitMQ и NATS [1], [2], [3], именно о них и пойдёт речь в данной работе.

Горизонтальное и вертикальное масштабирование

Для увеличения производительности в информационных системах можно выделать два типа масштабирования – вертикальное и горизонтальное [4].

Вертикальное масштабирование – это процесс увеличения ресурсов конкретного сервера (CPU, RAM, SSD). У такого типа масштабирования существует два критических недостатка: бесконечно наращивать мощности оборудования нельзя (мы ограничены техническими возможностями текущего поколения аппаратного обеспечения) и такой процесс связан с простоями, что может сильно сказаться на всех процессах компании. Вертикальное масштабирование может улучшить производительность только до новейших аппаратных возможностей. Этот вариант является недостаточным для технологических компаний с умеренной и высокой нагрузкой.

Горизонтальное масштабирование так же служит для увеличения производительности, только вместо увеличения ресурсов одного сервера, подключаются новые. При этом количество серверов, которые можно добавить в кластер, практически не ограничено. Это значительно дешевле, чем вертикальное масштабирование после определенного порога. Лучшее в горизонтальном масштабировании заключается в том, что у вас нет ограничений на масштаб масштабирования – когда производительность падает, просто добавьте еще одну машину, что возможно до бесконечности. Однако проблема в том, что не во всех системах возможно горизонтальное масштабирование.

Таким образом, более удобным является горизонтальное масштабирование. Однако, чтобы успешно его осуществить, требуется переход от монолитной архитектуры информационной системы к микросервисной. В качестве решения для такого перехода возможно использовать брокер данных, который будет выступать шиной данных, передающей данные между микросервисами.

Apache Kafka

Apache Kafka [5] является разработкой LinkedIn и использует схему обмена сообщениями «публикация-подписка» при помощи распределённого журнала фиксации сообщений. Он ориентирован на кластерную архитектуру, которая может использоваться

большим количеством сервисов-клиентов. Для реализации горизонтального масштабирования используется ZooKeeper, благодаря чему новые элементы кластера и потребители могут быть подключены без особых усилий. В отличие от других устойчивых очередей, которые обычно удаляют сохраненные сообщения при использовании, Kafka сохраняет их в течение настроенного периода времени. Это позволяет «воспроизвести» отправку в случае отказа потребителя. ZooKeeper упрощает управление кластерами Kafka, но при этом вводит еще один элемент, который необходимо поддерживать. Однако, Kafka предоставляет удобный API.

К основным достоинствам Kafka можно отнести:

1. Зрелое и большое сообщество;
2. Атомарность транзакций: операции чтения и записи в темах;
3. Журнал сообщений образует непрерывную последовательность – потребитель может легко найти последнее сообщение.

К недостаткам:

1. Потребитель не может подтвердить сообщение из другой цепочки;
2. Нет надежной репликации с несколькими контроллерами домена (в качестве решения можно использовать Confluent Enterprise);
3. Сложные механизмы управления.

RabbitMQ

RabbitMQ [6] – это механизм обмена сообщениями в соответствии с определением брокера AMQP 0-9-1. Advanced Message Queuing Protocol – это протокол обмена сообщениями, который позволяет соответствующим клиентским приложениям взаимодействовать с соответствующими брокерами промежуточного программного обеспечения обмена сообщениями. Он следует стандартному шаблону с промежуточным хранением, когда у вас есть возможность хранить данные в ОЗУ, на диске или в обоих типах памяти, поддерживает множество парадигм маршрутизации сообщений. RabbitMQ можно развернуть кластерно для повышения производительности и зеркально для обеспечения высокой доступности. Потребители обращаются прямо в очереди, а брокеры знают только об «обменах». Эти обмены связаны с очередями через привязки, которые определяют парадигму маршрутизации.

Очереди могут реплицироваться между узлами кластера, поэтому нет единой точки отказа или потери сообщений. RabbitMQ так же поддерживают синхронную и асинхронную отправку сообщений.

Достоинства:

1. Поддерживается репликация очередей на узлах кластера;
2. Повтор отправки сообщения до тех пор, пока не будет найден «живой» адресат;
3. Обширные инструменты маршрутизации.

Недостатки:

1. При присоединении к кластеру узлы отбрасывают свои данные (так называемые ситуации «split brain»);
2. Не гарантирует порядок доставки сообщений.
3. После доставки сообщение удаляется из системы.

NATS

NATS [7] изначально был построен на Ruby и достиг приличной скорости 150 тыс. сообщений в секунду. Однако после перехода на язык программирования Go, скорость возросла до 8-11 миллионов сообщений в секунду. Это образ Docker размером 3 МБ. NATS не поддерживает постоянный обмен сообщениями; если вы не в сети, вы не получите сообщения. Он работает как механизм публикации-подписки. NATS активно защищает себя и автоматически отсекает потребителей, которые не в сети или не успевают получать сообщения.

Среди пользователей NATS можно отметить следующие компании: Buzzfeed, Tinder, Stripe, Rakutan, Ericsson, HTC, Siemens, VMware, Pivotal, GE и Baidu.

Достоинства:

1. Слоган: «всегда включен и доступен»;
2. Низкое потребление аппаратных ресурсов;
3. Быстро: высокоскоростная коммуникационная шина;
4. Высокая масштабируемость;
5. Легкость: лишь 3-мегабайтный образ Docker.

Недостатки:

1. «Сработал и забыл»: NATS (нет поддержки постоянного обмена сообщениями);
2. Нет расширенных режимов доставки;
3. NATS не имеет репликации, сегментирования или полного упорядочивания. С NATS очереди сегментируются по узлам. Если узел умирает, его сообщения теряются. Входящие сообщения на активные узлы по-прежнему будут отправляться подключенным подписчикам, и ожидается, что подписчики повторно подключатся к пулу доступных узлов. После того, как ранее мертвый узел снова присоединится, он начнет получать сообщения.

Сравнение Nats и Apache Kafka

NATS был разработан для сценариев, в которых критически важны высокая производительность и низкая задержка, но при необходимости можно потерять некоторые данные, чтобы не отставать от данных – что в документации NATS описывается как «выстрелил и забыл». С архитектурной точки зрения это вызвано тем, что NATS не имеет уровня сохраняемости, который можно было бы использовать для долговременного хранения данных. В то время как у Apache Kafka есть уровень сохраняемости (с использованием хранилища в кластере).

Протокол Apache Kafka является двоичным по TCP, в отличие от NATS, являющегося простым текстом (также по TCP).

Шаблоны обмена сообщениями: оба поддерживают pub-sub и очереди, но NATS также поддерживает запрос-ответ (синхронизация и асинхронность).

NATS имеет концепцию очереди, и все подписчики, подключенные к одной очереди, в конечном итоге становятся частью одной и той же группы очередей.

Потоковая обработка: NATS не поддерживает потоковую обработку как первоклассную функцию, как Apache Kafka с Kafka Streams.

Клиенты Apache Kafka используют технику на основе опросов для извлечения сообщений, в отличие от NATS, когда сервер сам направляет сообщения клиентам (внутренне поддерживает граф интересов).

NATS может действовать довольно чувствительно в том смысле, что у него есть способность отключать потребителей, которые не успевают за темпами передачи данных, а также клиентов, которые не отвечают на запросы.

Проверка работоспособности потребителя также выполняется Apache Kafka. Этот процесс инициируется самим клиентом.

NATS, не имеет понятия о разделении /сегментировании сообщений, как это делает Apache Kafka.

В NATS нет внешней зависимости в случае NATS. Apache Kafka требует Zookeeper, что усложняет процесс настройки.

NATS в настоящее время не поддерживает репликацию (или какие-либо параметры высокой доступности). Это основной недостаток при сравнении с Apache Kafka.

Сравнение RabbitMQ и Apache Kafka

RabbitMQ имеет множество встроенных шаблонов обмена сообщениями [8]. Полнофункциональная маршрутизация избавляет получателей сообщений от необходимости извлечения, десериализации и проверки каждого сообщения, если им требуется только конкретное подмножество. Масштабирование производится при помощи добавления или удаления получателей. Также важным достоинством является архитектура, которая

позволяет поддерживать другие протоколы и осуществлять добавление нового функционала, например, консистентное хэширование.

Главным преимуществом Apache Kafka, благодаря относительным адресам получателей, является возможность «путешествия во времени» – просмотр уже отправленных сообщений, их повторная отправка и т.д.

Также Apache Kafka имеет возможность создания новых шаблонов путём сжатия журнала и сохранения данных, такой возможности нет у RabbitMQ.

Apache Kafka позволяет получить большее быстродействие от дисковой подсистемы, поскольку данные размещаются в памяти последовательно (sequential I/O), в отличие от RabbitMQ (random I/O) [9].

Более высокая производительность Apache Kafka обеспечивается и с помощью маршрутизации сообщений с одинаковым ключом одному и тому же адресату, что позволяет выполнять упорядоченную обработку с высокой степенью параллелизма.

Сравнение по техническим особенностям

1) Количество поддерживаемых платформ [10].

NATS: 48 типов клиентов. Серверы NATS могут быть скомпилированы на архитектурах Golang. Также NATS имеет двоичные дистрибутивы.

Apache Kafka: 18 типов клиентов. Серверы Kafka хорошо взаимодействуют с платформами под управлением Java.

RabbitMQ: более 50 типов клиентов. Поддерживаются серверы под управлением Linux и Windows.

2) Встроенные шаблоны и балансировка нагрузки.

NATS: Потоки реализованы с помощью встроенных шаблонов подписчиков для публикации/подписки, запроса/ответа и очереди с балансировкой нагрузки.

Apache Kafka: Потоки реализованы через публикацию/подписку. Балансировка нагрузки осуществляется с помощью разделения потребителей на группы. Приложение-потребитель должно соотносить запросы с ответами по нескольким темам для шаблона службы (запрос/ответ).

RabbitMQ: Потоки реализованы через публикацию /подписку и сервисы с функцией прямого ответа. Балансировка нагрузки осуществляется с помощью рабочей очереди. Приложения-потребители должны соотносить запросы с ответами по нескольким темам для шаблона службы (запрос/ответ).

3) Тип гарантии доставки данных.

Существует несколько основных типов доставки данных [11]:

1) at least once (хотя бы 1 раз): отправитель сообщения получает подтверждение от брокера, что гарантирует однократную запись сообщения. В случае ошибки или слишком долгой записи, есть возможность повторной отправки сообщения. Однако возможно дублирование сообщения в ситуации, когда аварийная ситуация возникла перед отправкой подтверждения о записи, но уже после самой записи сообщения.

2) at most once (не более 1-го раза): при возникновении ошибки или отсутствии подтверждения получения повторная отправка не выполняется. Однако при этом возможна ситуация, когда сообщение не было записано или получено адресатом. Это может привести к потере данных.

3) exactly once (строго однократно): сообщение в любом случае будет доставлено только один раз (даже в случае повторной отправки). В случае повторной отправки, сообщение всё равно будет записано в логе брокера только один раз. Данный тип гарантии доставки позволяет избавиться от потери данных и дублирования сообщений.

NATS: все три типа доставки.

Apache Kafka: exactly once, at least once

RabbitMQ: at most once, at least once.

4) Аутентификация.

NATS: поддерживает TLS, NKEYS (ключи NATS ED25519), аутентификацию по логину/паролю или сгенерированному токену. Установка ограничений для учетной записи: количество подключений, размер сообщения, количество импортированных / экспортированных данных, разрешения на публикацию и подписку, ограничения на подключение, ограничения адресов CIDR и ограничения по времени суток.

Apache Kafka: Поддерживает Kerberos и TLS. Поддерживает JAAS и готовую реализацию авторизатора (на основе ZooKeeper для хранения соединения и темы). Поддержка ACL для широкого набора ресурсов Kafka, включая темы, кластеры, группы и другие.

RabbitMQ: TLS, SASL, пара логин/пароль, а также подключаемая сторонняя авторизация. ACL определяют разрешения для операций настройки, записи и чтения для таких ресурсов, как обмены, очереди, транзакции и другие.

5) Сохранение и постоянство сообщений.

NATS: Поддерживает постоянство памяти, файлов и базы данных. Сообщения могут воспроизводиться по различным критериям: по времени, количеству или порядковому номеру. NATS сценарии могут архивировать старые сегменты журнала в холодное хранилище.

Apache Kafka: Поддерживает сохраняемость на основе файлов. Сообщения можно воспроизводить, указав смещение, и поддерживаются постоянные подписки. Поддерживается сжатие логов и KSQL.

RabbitMQ: Поддерживает сохраняемость на основе файлов. Поддерживает семантику на основе очереди, поэтому воспроизведение сообщений недоступно.

6) Доступность и отказоустойчивость.

NATS: Поддерживает кластеризацию с полной сеткой и функциями самовосстановления. У потоковой передачи NATS есть резервные серверы с возможностью горячего переключения.

Apache Kafka: Полностью реплицированные узлы кластера, координация происходит с помощью Zookeeper.

RabbitMQ: Поддержка кластеризации с полной репликацией данных. Однако кластеры нуждаются в сетях передачи данных с малой задержкой, в которых сетевые разделы встречаются редко.

7) Развертывание.

NATS: Сетевой элемент (сервер) NATS – это небольшой статический двоичный файл, что позволяет развёртывать его практически на любой платформе. NATS поддерживает архитектуру Adaptive Edge, которая позволяет выполнять крупные и гибкие развертывания. Отдельные серверы, конечные узлы и кластеры можно комбинировать любым способом для гибкого развертывания, подходящего для облачных, локальных, пограничных и IoT систем. Сервисы-клиенты не знают топологии и могут подключаться к любому серверу NATS.

Apache Kafka: Поддерживает кластеризацию с технологией зеркалирования для слабосвязанных удаленных кластеров. Клиенты привязаны к разделам в кластерах. Для серверов Kafka требуется JVM, CPU с 8 ядрами, от 64 ГБ до 128 ГБ ОЗУ, два или более дисков SAS/SSD на 8 ТБ и сетевая карта на 10 ГБ.

RabbitMQ: так же имеет кластерную архитектуру. Сервисы-клиенты не знают топологии и могут подключаться к любому кластеру. Также серверу требуется виртуальная машина Erlang.

8) Мониторинг.

NATS: Имеется возможность экспорта данных мониторинга в Prometheus и составление информационных панелей в Grafana. Также существует инструмент мониторинга разработки Nats-top.

Apache Kafka: имеет ряд инструментов для мониторинга: Confluent Control Center, Kafka Web Console, Kafka Offset Monitor, а также возможность подключения к сторонним сервисам мониторинга.

RabbitMQ: Инструменты CLI, система управления на основе плагинов с панелями мониторинга и сторонними инструментами.

9) Управление.

NATS: Управление правами доступа для пользователей может быть децентрализовано и выполняться через интерфейс командной строки. Конфигурация сервера отделена от настроек безопасности и выполняется так же с помощью командной строки и файла конфигурации, который может быть перезагружен с изменениями прямо во время выполнения.

Apache Kafka: Kafka имеет ряд инструментов управления и консолей для настроек безопасности и прав пользователей: Confluent Control Center, Kafka Web Console, Kafka Offset Monitor.

RabbitMQ: Имеет инструменты CLI, систему управления на основе плагинов с панелями мониторинга и поддержкой сторонних инструментов.

10) Интеграция.

NATS: Поддерживает связь со следующими сервисами: WebSockets, Kafka, IBM MQ, Redis, Apache Spark, Apache Flink, CoreOS, Elastic, Elasticsearch, Prometheus, Telegraf, Logrus, Fluent Bit, Fluentd, OpenFAAS.

Apache Kafka: имеет большое количество интеграций в своей экосистеме, включая потоковую обработку (Storm, Samza, Flink), Hadoop, базы данных (JDBC, Oracle Golden Gate), поиск и запросы (ElasticSearch, Hive), а также различные журналы и другие интеграции со сторонними сервисами.

RabbitMQ: имеет множество плагинов: протоколы (MQTT, STOMP), WebSockets, а также различные плагины авторизации и аутентификации.

Заключение

Все три представленных брокера данных являются высокопроизводительными решениями, которые могут быть применены в качестве шины данных и для дальнейшего перехода от вертикального к горизонтальному масштабированию. Однако в рамках телекоммуникационной компании имеются свои критерии.

NATS безусловно лидирует с точки зрения скорости передачи сообщений и лёгкости самого программного обеспечения. Однако он не поддерживает репликацию данных между элементами кластера, что может привести к потере данных – в рамках бизнес-процессов телекоммуникационной компании это является недопустимым. Также проблемой является и то, что если сервис-клиент не в сети, то он не получит сообщение. Это так же является проблемой.

RabbitMQ в свою очередь лишён этих недостатков. Однако он использует семантику на основе очереди, что делает недопустимым воспроизведение сообщений.

Таким образом, в рамках телекоммуникационной компании, где важна не только скорость передачи данных, но и надёжность с возможностью восстановления в случае аварии, оптимальным выбором является брокер данных Apache Kafka.

Библиографический список

1. Exploring Message Brokers: RabbitMQ, Kafka, ActiveMQ, and Kestrel [Электронный ресурс] URL: https://dzone.com/articles/exploring-message-brokers (дата обращения: 20.04.2021).
2. Best Message Queue (MQ) Software [Электронный ресурс] URL: https://www.g2.com/categories/message-queue-mq (дата обращения: 20.04.2021).
3. Modern Open Source Messaging: NATS, RabbitMQ, Apache Kafka, hmbdc, Synapse, NSQ and Pulsar [Электронный ресурс] URL: https://medium.com/@philipfeng/modern-open-source-messaging-apache-kafka-rabbitmq-nats-pulsar-and-nsq-ca3bf7422db5 (дата обращения: 20.04.2021).
4. Горизонтальное масштабирование. Что, зачем, когда и как? [Электронный ресурс] URL: https://habr.com/ru/company/oleg-bunin/blog/319526/ (дата обращения: 20.04.2021).

5. RabbitMQ против Kafka: два разных подхода к обмену сообщениями [Электронный ресурс] URL: https://habr.com/ru/company/itsumma/blog/416629/ (дата обращения: 20.04.2021).
6. RabbitMQ против Kafka: отказоустойчивость и высокая доступность в кластерах [Электронный ресурс] https://habr.com/ru/company/itsumma/blog/471858/ (дата обращения: 20.04.2021).
7. Выбор MQ для высоконагруженного проекта [Электронный ресурс] https://habr.com/ru/post/326880/ (дата обращения: 20.04.2021).
8. Kafka vs. RabbitMQ: Why Use Kafka? [Электронный ресурс] https://betterprogramming.pub/kafka-vs-rabbitmq-why-use-kafka-8401b2863b8b (дата обращения: 20.04.2021).
9. Apache Kafka и RabbitMQ: семантика и гарантия доставки сообщений [Электронный ресурс] https://habr.com/ru/company/itsumma/blog/437446/ (дата обращения: 20.04.2021).
10. Compare NATS [Электронный ресурс] https://docs.nats.io/compare-nats (дата обращения: 20.04.2021).
11. Что такое гарантия доставки сообщений или как избавиться от дублей и потерь в Apache Kafka и других Big Data брокерах [Электронный ресурс] https://www.bigdataschool.ru/blog/kafka-exactly-once.html (дата обращения: 20.04.2021).

SELECTING A DATA BROKER FOR USE AS A DATA-BUS IN A TELECOMMUNICATION COMPANY

Kalugin Vasiliy A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, vasyakalugin@mail.ru

Abstract. This paper compares data brokers for use as a data bus and further transitions from a monolithic information system architecture to a microservice-based architecture in a telecommunications company. The ways of increasing the productivity of the information system by means of horizontal and vertical scaling are considered. The horizontal scaling option was chosen as more versatile and affordable. Comparison of Apache Kafka, RabbitMQ and NATS data brokers is made. The advantages and disadvantages of each product are revealed, technical characteristics and features of operation are considered. Paired comparison of Apache Kafka and RabbitMQ, Apache Kafka and NATS. It is concluded that the most optimal product is Apache Kafka, since RabbitMQ and NATS do not meet the requirements for the reliability of data transmission within a telecommunications company.

Keywords: data broker, monolithic architecture, microservice architecture, scale-out, scale-out, Apache Kafka, RabbitMQ , NATS

УДК 004.021

СРАВНЕНИЕ СПОСОБОВ ИЗМЕРЕНИЯ СХОЖЕСТИ ПОЛЬЗОВАТЕЛЕЙ В МОДЕЛИ КОЛЛАБОРАТИВНОЙ ФИЛЬТРАЦИИ

Марьин Михаил Андреевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, muxaul.marin@email.ru

Рекомендательные системы, основанные на коллаборативной фильтрации, оценивают рейтинги пользователей, чтобы дать им более точные рекомендации. Одним из популярных способов прогнозирования рейтингов является использование моделей на основе ближайшего соседа, которые основываются на вычислении сходства между пользователями и используют концепцию, согласно которой похожие пользователи будут склонны оценивать одни и те же элементы одинаково. В этой статье представляется исследование наиболее часто используемых сходств (*PCS*, *CVS*, *MSD*, *SRC*, *FPC*, *WPC*) путем их использования на одном наборе данных и с учетом различных выборок из этого набора данных. Затем оцениваются предложенные меры сходства, используя одни и те же метрики качества, чтобы иметь объективную оценку и выбрать меру сходства, которая показывает наилучшую точность прогноза.

Ключевые слова: рекомендательные системы, коллаборативная фильтрация, близость векторов

I. Введение

В повседневной жизни мы обычно полагаемся на рекомендации других людей, компетентных в различных областях, предметах или продуктах. Вместо того, чтобы давать рекомендации ограниченному количеству людей, рекомендательные системы дают нам

возможность полагаться на мнение очень больших сообществ. Эти системы используют предпочтения людей, оценивают их и решают, какие персональные рекомендации им следует предоставить.

Существует несколько подходов к созданию рекомендаций, среди которых наиболее часто встречающимися являются следующие: основанные на метаинформации об объектах (*content-based*) [1] [2], демографические [3], основанные на знаниях и правилах (*rule-based, knowledge-based*) [4], использующие коллаборативную фильтрацию (*collaborative filtering*) [5] [6] [7] и гибридные подходы [8] [9].

Системы рекомендаций для коллаборативной фильтрации полагаются на рейтинги товаров, предоставленные пользователями. Эти рейтинги хранятся в матрице, где в строка является представлением пользователя, а столбец – представлением единицы контента. Система вычисляет рекомендации, используя отношения между пользователями и объектами, и прогнозирует оценки определенного пользователя по конкретным объектам. Наиболее часто используемые алгоритмы прогнозирования рейтингов – это модели на основе ближайшего соседа и модели скрытых переменных [10].

В этой статье рассматриваются ориентированные на пользователей алгоритмы коллаборативной фильтрации, которые вычисляют прогнозируемые рейтинги для целевого пользователя на основе рейтингов других похожих пользователей, то есть пользователей, которые имеют похожие оценки. Такие рейтинги используются, чтобы давать рекомендации этому целевому пользователю. Для этого необходимо рассчитать сходство между пользователями. Существует несколько формул, которые используются, но наиболее популярной является формула корреляции Пирсона (PCS), которая стала стандартным способом вычисления корреляции. В этой статье проведены эксперименты для проверки качества наиболее часто используемых мер сходства, четкое сравнение между ними с одним и тем же набором данных, и используются одни и те же метрики качества для оценки.

II. Цели и задачи исследования

Для создания рекомендательного сервиса внутри онлайн-кинотеатра Movix.ru необходимо найти наилучший способ расчета близости векторов пользователей в задаче коллаборативной фильтрации. Поиск меры близости – цель данной работы.

Для достижения поставленной цели необходимо решить ряд задач, а именно: описать способы расчета близости векторов, провести вычислительные эксперименты, провести сравнительный анализ результатов.

III. Меры сходства.

A. Pearson Correlation Similarity (PCS)

Это популярный способ сравнения рейтингов. Для пользовательского алгоритма корреляция Пирсона [7] между пользователями *u* и *v* равна:

$$PCS(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \overline{r}_u)^2 \sum_{i \in I_{uv}} (r_{vi} - \overline{r}_v)^2}}$$

Где *u* – пользователь *u*, *v* – пользователь *v*, I_{uv} – список элементов, оцененных как пользователями *u*, так и *v*, r_{ui} – рейтинг пользователя u по элементу I, а $\overline{r}_u$ – среднее значение оценок, предоставленных пользователем *u*.

Основным недостатком этой меры сходства является то, что она не дает точного результата, если у двух пользователей один общий подход к оцениванию или когда определенный пользователь оценил только один элемент.

B. Cosine Vector Similarity (CVS)

CVS между двумя пользователями *u* и *v* вычисляется следующим образом [7]:

$$CVS(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2 \sum_{i \in I_{uv}} r_{vi}^2}}$$

где I_{uv} обозначает набор элементов, оцененных как *u*, так и *v*. Неизвестные оценки считаются равными 0, это приводит к тому, что они фактически выпадают из числителя. Если среднее значение рейтинга по пользователю вычтено из оценок перед вычислением сходства, *CVS* будет эквивалентен *PCS* [13].

Недостатком этой меры является то, что она не учитывает различия в среднем и дисперсии оценок, выставленных пользователями *u* и *v* [7]. Например, если один пользователь оценивает пять фильмов как 1 из 5 звезд, а другой пользователь оценивает те же пять фильмов как 5 из 5 звезд, их *CVS* будет равен 1, что кажется неточным показателем сходства пользователей.

C. Mean Squared Difference (MSD)

Оценивает сходство между двумя пользователями *u* и *v* как величину, обратную среднему квадрату разницы между оценками *u* и *v* по одним и тем же пунктам [7]. *MSD* уделяет больше внимания большим различиям между оценками пользователей, чем небольшим различиям.

$$MSD(u,v) = \frac{|I_{uv}|}{\sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2}$$

Среднее значение этих квадратов разностей тогда обеспечило бы меру сходства, чем меньше среднеквадратичная разница, тем больше сходство.

Его недостатком является то, что он ограничен по сравнению с *PCS*, поскольку не фиксирует отрицательную корреляцию между предпочтениями пользователя. Наличие таких отрицательных корреляций может повысить точность прогнозирования рейтинга [7].

D. Frequency-weighted Pearson Correlation (FPC)

Рейтинги элементов с большим разбросом оценок более информативны, чем те, которые имеют общие оценки. Например, рассмотрим фильм, который нравится большинству людей. Если вычисление веса основано на таких фильмах, будут получены искусственно завышенные значения. Аналогичным образом, пользователь, который всегда имеет один и тот же подход к выставлению рейтинга, не будет предоставлять системе ценную информацию по сравнению с пользователем, чьи предпочтения варьируются от одного элемента к другому. Подход к решению этой проблемы заключается в присвоении веса λ_i каждому элементу *i* следующим образом:

$$\lambda_i = \log \frac{|U|}{|U_i|}$$

Где *U* – это набор всех пользователей, а U_i – это набор пользователей, которые оценили элемент *i*. Чтобы вычислить *FPC* между пользователями *u* и *v*, корреляция между рейтингами, присвоенными элементу *i*, взвешивается как λ_i: [7]

$$FPC(u,v) = \frac{\sum_{i \in I_{uv}} \lambda_i (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i \in I_{uv}} \lambda_i (r_{ui} - \overline{r}_u)^2 \sum_{i \in I_{uv}} (r_{vi} - \overline{r}_v)^2}}$$

Это снижает влияние элементов, имеющих много оценок, на корреляцию Пирсона. Если все пользователи оценили элемент *i*, то λ_i будет равно нулю. Этот подход улучшает точность прогнозирования пользовательской рекомендации [6].

E. Weighted-Pearson Correlation (WPC)

Во многих случаях некоторые популярные фильмы могут иметь рейтинги, выставленные слишком большим количеством пользователей. Эти рейтинги могут ухудшить качество рекомендаций, потому что они не сильно различают пользователей. Поэтому предлагаемое решение состоит в модификации *PCS* с учетом веса [11]. Оно похоже на *FPC*, но средний рейтинг μ_u вычисляется для каждого пользователя *u* с использованием указанных рейтингов следующим образом:

$$WPC = \frac{\sum_{j \in I_u \cap I_v} w_j (r_{uj} - \mu_u)(r_{vj} - \mu_v)}{\sqrt{\sum_{j \in I_u \cap I_v} w_j (r_{uj} - \mu_u)^2} \sqrt{\sum_{j \in I_u \cap I_v} w_j (r_{vj} - \mu_v)^2}}, \text{ где } \mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|}$$

Довольно часто (и это проще в вычислительном отношении) вычислять каждый средний рейтинг μ_u только один раз для каждого пользователя *u*. I_u – это подмножество элементов, оцененных пользователем *u*

$$w_k = \log(m / m_k)$$

Где m_k – количество оценок элемента *k*, а *m* – общее количество пользователей.

F. Spearman Rank Correlation (SRC)

В то время как *PCS* напрямую использует рейтинговые значения, *SRC* вместо этого рассматривает ранжирование рейтингов. Элементы, оцененные пользователем, ранжируются таким образом, что элемент с наивысшим рейтингом имеет ранг 1, а элементы с более низким рейтингом имеют более высокие ранги. Элементам, имеющим одинаковый рейтинг, присваивается средний рейтинг для их позиции [12]. Сходство *SRC* между пользователями *u* и *v* оценивается как:

$$SRC(u,v) = \frac{\sum_{i \in I_{uv}} (k_{ui} - \bar{k}_u)(k_{vi} - \bar{k}_v)}{\sqrt{\sum_{i \in I_{uv}} (k_{ui} - \bar{k}_u)^2 \sum_{i \in I_{uv}} (k_{vi} - \bar{k}_v)^2}}$$

где k_{ui} – рейтинг элемента *i* в списке рейтинга пользователя *u*, а $\bar{k}_u$ - средний рейтинг элементов, оцененных *u*.

Хотя эта мера позволяет избежать проблемы нормализации рейтинга с помощью ранжирования, это нехорошо, когда диапазон рейтингов имеет только несколько возможных значений, так как это создаст большое количество связанных оценок.

IV. Исследования в данной области

В предыдущих исследованиях сравнивались характеристики сходства *MSD*, корреляций Пирсона и Спирмена с использованием оценок с сайта рекомендаций фильмов *MovieLens* [13]. Используемый метод заключался в том, что список кандидатов-соседей оценивал элементы. Прогноз новых рейтингов был сделан с помощью модели *k* соседей, вес подобия которых имеет наибольшую величину. Результаты приведены для разных значений *k*. Было замечено, что *MSD* дает наименее точные прогнозы, в то время как *PCS* имеет большую точность, чем *SRC*. *PCS* была признана лучшей мерой сходства, хотя более недавнее исследование показало, что эффективность этой меры сильно зависит от данных.

В другом исследовании [14] несколько показателей сходства сравнивались с использованием набора данных, состоящего из 122 176 оценок. 10% пользователей были случайным образом выбраны в качестве тестовых пользователей. Было показано, что корреляция Спирмена и Пирсона дает очень близкие результаты точности с использованием средней абсолютной ошибки (*Mean Absolute Error*). *CVS* показала себя успешным в поиске информации, но не работает так же хорошо, как *PCS* в коллаборативной фильтрации. Кроме того, в статье был сделан вывод, если шкала оценок состоит из небольшого числа дискретных рангов, рекомендуется использовать *SRC*.

V. Сравнение мер сходства

В этом разделе представляется реализации: используемый язык программирования, а также набор данных и метрики для оценки. Несмотря на то, что *PCS* обычно рассматривается как лучший показатель сходства по сравнению с другими мерами сходств в ранее замеченных исследованиях, общего исследования характеристик других сходств не проводилось. Поэтому в этом разделе протестировались наиболее часто используемые меры сходства и сравнили их эффективность и точность.

A. Метрики качества

Для измерения точности прогнозов и анализа, насколько хорошо система генерирует прогнозы, были выбраны: mAP@k (Mean Average Precision at k), nDCG@k (Normalized Discounted Cumulative Gain as k), Recall@k (Mean Recall at k). Первые две метрики

учитывают ранжирование элементов, а Recall@k учитывает только вхождение True Positive элементов в список прогнозируемых элементов, поэтому будет наблюдаться существенное различие между двумя группами метрик.

B. Моделирование

В этом исследовании использовались данные онлайн кинотеатра Movix за 2020 год.

Набор данных содержит 1 миллион уникальных пользователей и 152456 контента, которым было выставлено 3 124 589 оценок. Зафиксировано 213 778 965 фактов просмотров единиц контента, 65 123 657 – поисковых запросов, 947 659 – добавления в закладки или избранное, 18 631 478 – просмотров трейлеров к фильмам.

Чтобы проверить результаты, изначально были отделены четыре подмножества этого набора данных, имеющих размеры соответственно 0,5%, 1%, 5% и 10% по отношению к размеру полного набора данных. Также использовали пять различных версий исходного набора данных, чтобы иметь возможность проверить, связаны ли характеристики сходства со структурой набора данных или нет. Среднее значение показателей было рассчитано для всех этих наборов данных, чтобы обеспечить более реалистичные и точные результаты.

C. Результаты исследования

Ниже представлено сравнение между изученными мерами сходства. Начинаем с оценки эффективности 6 сходств с использованием показателей точности прогноза (*nDCG@k*, *mAP@k*, *Recall@k*). Оценки качества *nDCG@k* и *mAP@k* учитывают порядок рекомендованных элементов, а *Recall@k*- нет. В связи с этим первые 2 метрики качества на порядок ниже. Так же стоит отметить, что идеальные веса в *nDCG@k* создавались линейно из-за чего *nDCG@k* и *mAP@k* имеют достаточно сильную корреляцию. Это показано на рисунках 1, 2 и 3 и в таблице 1.

На рисунке 1 изображено использование *nDCG@k* при проверке мер сходства на первых 10, 50, 100 и 200 рекомендованных элементах. Все меры сходства имеют почти одинаковую производительность, кроме *MSD,* которая существенно просела по качеству, и *CVS*, которая сильно выигрывает по качеству у остальных мер сходства. Только при рекомендации 200 элементов *FPC* имеет приблизительно такую же точность.

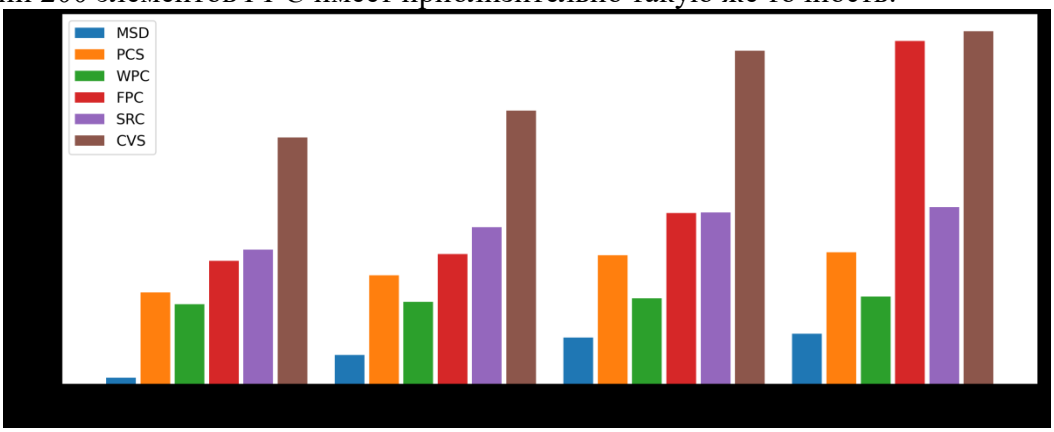


Рис. 1. Оценка качества моделей по *nDCG@k*

На рисунке 2 изображено использование *mAP@k* в качестве меры качества модели. В целом ситуация аналогична с предыдущей метрикой, за исключением меньшего отставания *FPC* и *SRC* от *CVS* при отборе 100 и 200 рекомендаций.

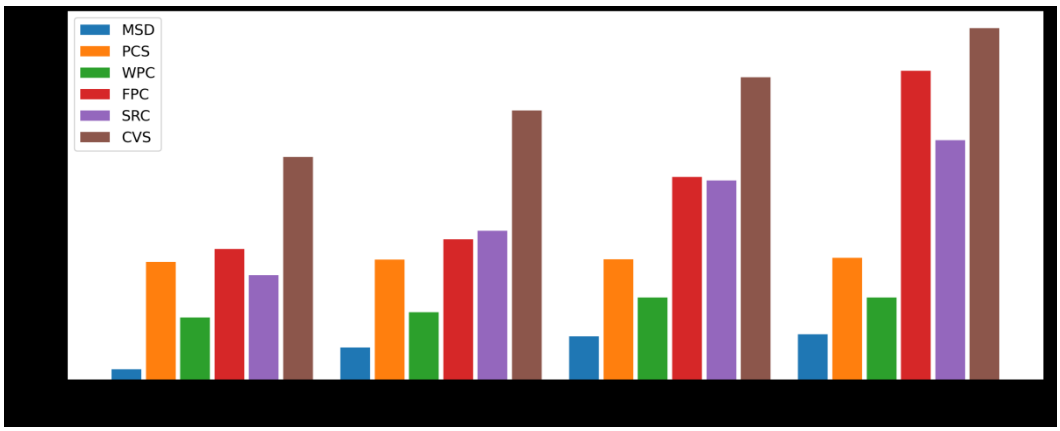


Рис. 2. Оценка качества моделей по mAP@k

На рисунке 3 изображено использование *Recall@k* в качестве меры качества модели. Большинство моделей имеют приблизительно одинаковые результаты. Это говорит о том, что различные меры близости хорошо отбирают рекомендации, но плохо ранжируют их по релевантности.

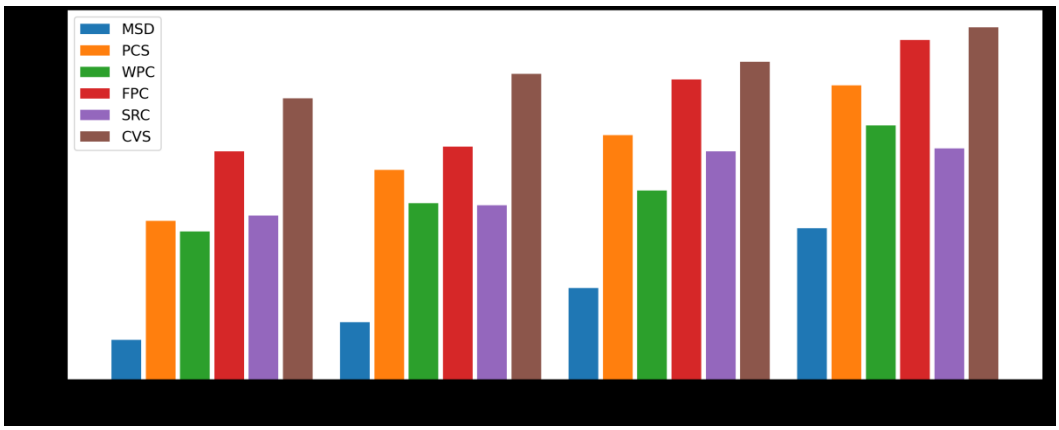


Рис. 3. Оценка качества моделей по Recall@k

Таблица 1. Результаты исследования мер сходства

	Pearson Correlation Similarity				Spearman Rank Correlation			
k	10	50	100	200	10	50	100	200
NDCG@k	0,0215	0,0255	0,0302	0,0309	0,0315	0,0368	0,0402	0,0415
mAP@k	0,0289	0,0295	0,0296	0,0299	0,0257	0,0366	0,0489	0,0588
Recall@k	0,1022	0,1351	0,1574	0,1895	0,1056	0,1123	0,1469	0,1489
	Frequency-weighted Pearson Correlation				Cosine Vector Similarity			
k	10	50	100	200	10	50	100	200
NDCG@k	0,0289	0,0305	0,0401	0,0804	0,0578	0,0641	0,0781	0,0827
mAP@k	0,0321	0,0345	0,0498	0,0758	0,0547	0,0661	0,0742	0,0863
Recall@k	0,1469	0,1501	0,1933	0,2187	0,1811	0,1968	0,2047	0,2269
	Weighted-Pearson Correlation				Mean Squared Difference			
k	10	50	100	200	10	50	100	200
NDCG@k	0,0187	0,0193	0,0201	0,0205	0,0015	0,0068	0,0109	0,0118
mAP@k	0,0153	0,0166	0,0202	0,0202	0,0026	0,0079	0,0107	0,0112
Recall@k	0,0954	0,1136	0,1217	0,1636	0,0256	0,0369	0,0589	0,0974

VI. Заключение

В этой статье сравнивались шесть мер сходства, используемых в коллаборативной фильтрации на основе соседства (корреляция Пирсона, косинусный вектор, частотно-взвешенная корреляция Пирсона, взвешенная корреляция Пирсона, ранговая корреляция Спирмена, сходство среднеквадратичной разницы) с тремя метриками: *nDCG@k* и mAP@k и Recall@k. Исследование проводилось на наборе данных онлайн-кинотеатра Movix.ru, и результаты сравнения были проанализированы.

Список литературы

1. P. Lops, M. De Gemmis and G. Semeraro, "Content-Based Recommender Systems: State Of The Art And Trends",Recommender Systems Handbook, Springer, 2011, pp. 73–105.
2. R.Van Meteren and S. Van Someren, "Using Content-Based Filtering For Recommendation", 2000.
3. L.Safoury and A. Salah, "Exploiting User Demographic Attributes For Solving Cold-Start Problem In Recommender System",. Lecture Notes on Software Engineering, Vol. 1, No. 3, 2013.
4. R. Burke, "Knowledge-Based Recommender Systems", Encyclopedia of Library and Information Systems, Vol. 69, Issue 32, pp. 175-186, 2000.
5. Lee, Yunkyoung, "Recommendation System Using Collaborative Filtering" (2015). Master's Projects. 439. Http://Scholarworks.Sjsu.Edu/Etd_Projects/439
6. Y. Koren and R. Bell, "Advances In Collaborative Filtering", Recommender Systems Handbook, Springer, 2015, pp. 77-118.
7. C. Desrosiers and G. Karypis, "A Comprehensive Survey Of Neighborhood-Based Recommendation Methods", Recommender Systems Handbook, Springer, 2011, pp. 107–144.
8. R. Burke, "Hybrid Recommender Systems: Survey And Experiments", User Modeling and User-Adapted Interaction Journal, Vol. 12, Issue 4, pp. 331–370, 2002.
9. M. Degemmis, P. Lops and G. Semeraro, "A Content-Collaborative Recommender That Exploits Wordnet-Based User Profiles For Neighborhood Formation", User Model and User-Adapted Interaction Journal, Vol. 17, Issue 3, pp. 217–255 2007.
10. M. Elahi, F. Ricci and N. Rubens, "A Survey Of Active Learning In Collaborative Filtering Recommender Systems", Elsevier Computer Science Review, Vol. 20, pp. 29-50 2016.
11. C. C. Aggrwal, "Recommender Systems", Textbook, pp. 29-70, 2016.
12. M. D. Ekstrand, J. T. Riedl and J. A. Konstan, "Collaborative Filtering Recommender Systems", Journal Foundations and Trends in Human-Computer Interaction, Volume 4 Issue 2, February 2011, Pages 81-173.
13. J. Herlocker, J. A. Konstan and J. Riedl, "An Empirical Analysis Of Design Choices In Neighborhood-Based Collaborative Filtering Algorithms", 2002.
14. J. L. Herlocker, J. A. Konstan, A. Borchers and J. Riedl, "An Algorithmic Framework For Performing Collaborative Filtering", 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 230-237, 1999.

COMPARISON OF WAYS TO MEASURE USER SIMILARITY IN THE COLLABORATIVE FILTERING MODEL

Mar'in Mikhail Andreevich

Perm State University, 15, Bukireva st., Perm, 614990, Russia, muxaul.marin@email.ru

Abstract. Collaborative filtering-based recommender systems evaluate user ratings to give users more accurate recommendations. One popular way to predict ratings is to use nearest-neighbor models, which are based on calculating similarity between users and use the concept that similar users will tend to rate the same items the same way. This paper presents a study of the most commonly used similarities (PCS, CVS, MSD, SRC, FPC, WPC) by using them on a single data set and considering different samples from that data set. The proposed similarity measures are then evaluated using the same quality metrics in order to have an unbiased evaluation and to select the similarity measure that shows the best prediction accuracy.

Key words: recommendation systems, collaborative filtering, vector similarity.

УДК 004.733, 681.518.5

ВОЗМОЖНОСТЬ ИСПОЛЬЗОВАНИЯ ДАТЧИКОВ ДЛЯ АВТОМАТИЧЕСКОГО НАБЛЮДЕНИЯ ЗА СОСТОЯНИЕМ КИЗЕЛОВСКОГО УГОЛЬНОГО БАССЕЙНА

Никитина Елена Юрьевна, Свирепова Анна Андреевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, ann.svirepova@yandex.ru

Рассматривается проблема загрязнения сточных вод Кизеловского угольного бассейна. На данный момент анализ сточных вод КУБа проводится специалистами вручную и очень редко. Для автоматизации процесса сбора показателей сточных вод необходимо произвести выбор подходящих датчиков. Приведена классификация существующих датчиков. Произведен выбор типа датчиков, которые подходят для автоматизации процесса сбора показателей сточных вод Кизеловского угольного бассейна. Рассмотрены датчики мониторинга качества воды таких компаний как Libelium и Systemelektronik. Приведена сравнительная характеристика предложенных датчиков. Сравнение произведено по четырем критериям: параметры, которые могут измерять датчики, способы передачи данных датчиков, возможность масштабируемости системы датчиков, возможность удаленного управления датчиками. На основе составленной характеристики был произведен выбор в пользу одной из рассмотренных систем датчиков.

Ключевые слова: Кизеловский угольный бассейн, автоматизация процесса сбора показателей, датчики мониторинга качества воды.

Кизеловский угольный бассейн (КУБ) расположен в восточной части Пермского края. В административном отношении бассейн находится в пределах Кизеловского, Гремячинского и Чусовского муниципальных районов и Губахинского городского округа Пермского края. В пределах границ бассейна находятся крупные населенные пункты. [1]

В границы КУБа попадают особо охраняемые природные территории (ООПТ) регионального значения – 11 памятников природы и один охраняемый Ландшафт.

Несмотря на то, что с 2002 года прекращена добыча угля в Кизеловском угольном бассейне, он негативно влияет на территорию Пермского края. Негативное влияние распространяется значительно дальше границ бассейна. Основную роль в этом играют реки, которые переносят загрязнители на большие расстояния. Загрязнению также подвержено Камское водохранилище.

Для сточных вод основными загрязняющими веществами являются органические соединения, взвешенные вещества, нефтепродукты, минеральные соли, в том числе тяжелых металлов. При очистке в прудах-отстойниках, прудах-осветителях скапливаются взвешенные частицы и компоненты, осаждающиеся из раствора. Неочищенные и частично очищенные воды сбрасываются в гидросеть, фильтруются в подземные воды и, таким образом, являются важным фактором миграции веществ в угледобывающих районах. Сточные воды могут оказывать негативное влияние на окружающую среду и после прекращения разработок угля. Исследования на территории угольных бассейнов показывают, что закрытие шахт в ряде случаев вызывает ухудшение экологической обстановки в результате неконтролируемого поступления загрязненных подземных вод на поверхность.

На данный момент анализ сточных вод КУБа проводится специалистами вручную и очень редко. Для автоматизации процесса анализа необходимо обеспечить сбор показателей сточных вод, для чего в первую очередь необходимо выбрать подходящие датчики. Далее будет разработана крупная информационная система для наблюдения за состоянием КУБа.

В настоящее время для мониторинга в Кизеловском угольном бассейне применяется информационная система ГИС КУБ. Это картографический веб-сервис, который расположен по адресу http://kub.maps.psu.ru/. [2]

Картографический веб-сервис представляет собой клиентское вебприложение, реализованное с помощью технологий HTML5, CSS3 и JavaScript, для его публикации используется веб-сервер Apache HTTP 2.4. Для публикации векторных пространственных данных на веб-сервер использован программный продукт ArcGIS Server 10.4.1. Основной функционал картографического веб-сервиса реализован при помощи библиотеки ArcGIS API for JavaScript.

Для источников загрязнения (изливов кислых шахтных вод, отвалов, родников), а также для пунктов наблюдения за поверхностными и подземными водами, присутствует возможность визуализации результатов измерений в виде столбчатых диаграмм многолетних данных (с 2006 по 2017 гг.) о химическом составе вод, расходе (для изливов) и глубине их залегания (для пунктов наблюдения за подземными водами). [3]

Вышеописанная ИС должна быть расширена модулем автоматического сбора измерений загрязнения КУБа и агрегированием полученных данных в и информационные ресурсы ГИС КУБ. На первом этапе этой разработки необходимо определиться с выбором датчиков, которые позволят автоматически получать данные о различных параметрах загрязнения КУБа.

В настоящее время датчики реализуются на основе различных физических явлений и различных технологий. Это дает возможность делать их как многофункциональными, т. е. обеспечивающими посредством одного датчика преобразование многих физических величин, так и "интеллектуальными", позволяющими выявлять при преобразовании определенной величины воздействия других величин, которые искажают результат преобразования.

Датчик – это устройство, которое обнаруживает изменения в окружающей среде и передает сигналы о них на выходные каналы другой системы. Датчик переводит физическое явление в измеряемое аналоговое напряжение (или цифровой сигнал), преобразованное в доступную для чтения форму или передаваемое для чтения и дальнейшей обработки. [4]

Разные датчики предназначены для измерения разных физических явлений:

- Термопары, РДТ и термисторы: для измерения температуры
- Тензодатчики: для измерения деформации объекта, например, давления, натяжения, веса и пр.
- Датчики нагрузки: для измерения веса и нагрузки

- Датчики LVDT: для изучения смещения в расстоянии
- Акселерометры: для измерения вибрации и ударных нагрузок
- Микрофоны: для регистрации звуковых волн
- Преобразователи тока: для измерения переменного и постоянного тока
- Трансформаторы напряжения: для измерения потенциалов высокого напряжения
- Оптические датчики: для обнаружения света, передачи данных и замены традиционных датчиков
- Датчики-видеокамеры: для захвата одного непрерывного двумерного изображения
- Цифровые датчики: для подсчета дискретных значений, линейного и ротационного кодирования, измерения положения и пр.
- Датчики местонахождения (GPS): для записи географических координат на основе данных GPS, ГЛОНАСС и других спутниковых систем позиционирования. Разные датчики GPS обеспечивают разную точность определения местоположения и многие другие
- Датчики мониторинга качества воды: предназначены для мониторинга качества питьевой воды, управления рыбоводческими хозяйствами, обнаружения утечек химических веществ, удаленного анализа воды в бассейнах и спа, а также контроля загрязнения морской воды.

Рассмотрим подробнее разные датчики мониторинга качества воды.

Libelium Smart Water Xtreme

Компания Libelium разработала IoT-платформу Smart Water Xtreme, см. рис. 1. Это решение содержит цифровые и оптические датчики и обладает непревзойденной надежностью, прочностью и устойчивостью к изменениям условий окружающей среды. Платформа измеряет следующие показатели: содержание растворенного кислорода, pH, окислительно-восстановительный потенциал (ОВП), проводимость, минерализация, общая минерализация (TDS – KCl), температура, нефелометрическая мутность, взвешенные твердые частицы и взвешенный слой осадка. Также в платформу добавлены новые ионоселективные датчики для обнаружения аммиака, нитратов, хлоридов, натрия и кальция. [5]

Рис. 1- IoT-платформа Smart Water Xtreme

Датчики используют протоколы RS-485 и SDI-12; максимальная длина кабеля составляет 50 метров. На платформу можно установить датчики со степенью защиты IP68,

которые могут работать под водой при давлении до 5 бар (глубина 50 м). Это позволяет использовать платформу для изучения водяного столба – концепции, которая используется в океанографии для описания физических и химических характеристик морской воды на различных глубинах. Таким образом, теперь можно устанавливать несколько датчиков для измерения определенного параметра (например, содержания растворенного кислорода) на различной глубине.

Libelium Smart Water

Платформа беспроводных датчиков Smart Water для упрощения удаленного мониторинга качества воды. Waspmote Smart Water, оснащенная несколькими датчиками, которые измеряют десяток наиболее важных параметров качества воды, имеет автономные узлы, которые подключаются к облаку для контроля воды в реальном времени. Waspmote Smart Water подходит для мониторинга питьевой воды, обнаружения утечек химикатов в реках, дистанционного измерения бассейнов и спа, а также уровней загрязнения морской воды. Измеряемые параметры качества воды включают pH, растворенный кислород (DO), окислительно-восстановительный потенциал (ОВП), проводимость (соленость), мутность, температуру ирастворенные ионы (фторид (фторид (F –), кальций (Ca 2+), нитрат (NO 3 –), хлорид (Cl –), йодид (I –), медь (Cu 2+), бромид (Br –), Серебро (Ag +), фторборат (BF 4 –), аммиак (NH 4), литий (Li +), магний (Mg 2+), нитрит (NO 2-), перхлорат (ClO 4), калий (K +) , Натрий (Na +). [6]

Платформа Waspmote Smart Water – это сенсорный узел со сверхнизким энергопотреблением, предназначенный для использования в суровых условиях и развертывания в умных городах в труднодоступных местах для обнаружения изменений и потенциального риска для здоровья населения в режиме реального времени.

Рис. 2 – Платформа беспроводных датчиков Smart Water

Waspmote может использовать сотовую связь (3G, GPRS, WCDMA) и связь 802.15.4 / ZigBee (868/900 МГц) с большим радиусом действия для отправки информации в облако, а также может использовать солнечные панели, которые заряжают аккумулятор для поддержания автономности. Узлы Smart Water готовы к развертыванию «из коробки», а датчики можно откалибровать или заменить в полевых условиях с помощью комплектов, предоставленных Libelium.

GO Systemelektronik GmbH BlueBox System

Системы непрерывного измерения концентрации различных загрязняющих веществ в сбросах (стоках) на основе датчиков, устанавливаемых в поток. [7]

Рис. 3 – Система измерения и управления BlueBox

Система измерения и управления BlueBox является модульной и масштабируемой базой для реализации проектов всех размеров. Она позволяет управлять сотнями датчиков, на основе которых строятся анализаторы воды, и исполнительных механизмов, выполнять самые сложные функции управления, контроля и автоматизации и обеспечивает надежный мониторинг.

Передача данных и событий может осуществляться через сети и системы мобильной телефонной связи, при этом программные продукты, обеспечивающие визуализацию и контроль, работают на всех уровнях сбора и обработки информации. Кроме использования возможностей автономной работы с использованием встроенного цветного сенсорного экрана, все многоканальные анализаторы и другие компоненты семейства BlueBox предназначены для подключения на основе протокола TCP / IP с использованием различных средств связи, включая стационарные, мобильные и спутниковые системы. Ряд функций системы, включая визуализацию и управление, доступны для удалённого управления из специализированных диспетчерских центров. Среди других возможностей системы – оповещения через службу коротких сообщений, факс и электронную почту, а также безопасная передача данных во внешние информационные базы.

Датчики, используемые в системах непрерывного контроля

- Специализированные сенсоры фирмы для определения содержание в стоках следующих веществ: аммоний (NH4+); бромид; кальций; хлориды (соли соляной кислоты); фториды; нитраты; калий.
- Флуоресцентный датчик фикоцианина, хлорофилла, растворённых органических веществ, нефтепродуктов.
- Электрохимический датчик хлорина, диоксида хлора, озона.
- Кислородно – температурный сенсор.
- Стеклянный электрод для определения водородного показателя pH.

Кроме сенсоров, осуществляющих контроль сточных вод, которые реагируют на наличие в потоке химических соединений, GO Systemelektronik GmbH предлагает широкий выбор датчиков, контролирующих различные физические параметры: атмосферные осадки; солнечное излучение; давление воды; направление ветра, его скорость; 10 разновидностей температурных сенсоров; 2 вида приборов, оценивающих мутность воды; проверка уровня электрической проводимости среды.

Сравнительная таблица

	Libelium Smart Water Xtreme	**Libelium Smart Water**	**GO Systemelektronik GmbH BlueBox System**
Измеряемые параметры	содержание растворенного кислорода, pH, окислительно-восстановительный	pH, растворенный кислород (DO), окислительно-восстановительный потенциал (ОВП),	Специализированные сенсоры фирмы определяют содержание в стоках следующих веществ:

	Libelium Smart Water Xtreme	**Libelium Smart Water**	**GO Systemelektronik GmbH BlueBox System**
	потенциал (ОВП), проводимость, минерализация, общая минерализация (TDS – KCl), температура, нефелометрическая мутность, взвешенные твердые частицы и взвешенный слой осадка, датчики обнаружения аммиака, нитратов, хлоридов, натрия и кальция	проводимость (соленость), мутность, температуру растворенные ионы (фторид (фторид (F –), кальций (Ca 2+), нитрат (NO 3 –), хлорид (Cl –), йодид (I –), медь (Cu 2+), бромид (Br –), Серебро (Ag +), фторборат (BF 4 –), аммиак (NH 4), литий (Li +), магний (Mg 2+), нитрит (NO 2-), перхлорат (ClO 4), калий (K +) , Натрий (Na +)	аммоний (NH4+); бромид; кальций; хлориды (соли соляной кислоты); фториды; нитраты; калий. Флуоресцентный датчик фикоцианина, хлорофилла, растворённых органических веществ, нефтепродуктов. Электрохимический датчик хлорина, диоксида хлора, озона. Кислородно – температурный сенсор. Стеклянный электрод для определения водородного показателя pH.
Передача данных	протоколы RS-485 и SDI-12; максимальная длина кабеля составляет 50 метров.	сотовая связь (3G, GPRS, WCDMA) и связь 802.15.4 / ZigBee (868/900 МГц) с большим радиусом действия для отправки информации в облако	Передача данных и событий может осуществляться через сети и системы мобильной телефонной связи, при этом программные продукты, обеспечивающие визуализацию и контроль, работают на всех уровнях сбора и обработки информации.
Масштабируемость	Нет	Нет	Да
Удаленное управление	Да	Да	Да

Из вышеперечисленных датчиков для решения поставленной задачи более всего подходит платформа Libelium Smart Water. Она является беспроводной и имеет возможность измерения всех необходимых параметров.

Библиографический список

1. *Н. Г. Максимович и С. В. Пьянков*, Кизеловский угольный бассейн:, Пермь: «Раритет-Пермь», 2018.
2. *О. А. Березина*, КОМПЛЕКСНАЯ ОЦЕНКА ГЕОЭКОЛОГИЧЕСКИХ ПОСЛЕДСТВИЙ, Пермь: на правах рукописи, 2019.
3. *Р. К. Абдуллин, О. А. Березина и Н. Г. Максимович*, «СОЗДАНИЕ БАЗЫ ГЕОДАННЫХ ДЛЯ ОЦЕНКИ СОСТОЯНИЯ ОКРУЖАЮЩЕЙ СРЕДЫ ЛИКВИДИРОВАННОГО КИЗЕЛОВСКОГО УГОЛЬНОГО БАССЕЙНА,» *ИЗВЕСТИЯ ТУЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА,* № 3, pp. 3-17, 2018.
4. «Компания DEWESoft,» [В Интернете]. Available: https://dewesoft.com/ru/daq/what-is-a-sensor. [Дата обращения: 09 04 2021].
5. «Компания ООО «СМАРТ Дистрибьюшн»,» [В Интернете]. Available: https://iotsmart.ru. [Дата обращения: 09 04 2021].
6. «Компания Libelium,» [В Интернете]. Available: https://www.libelium.com/libeliumworld/smart-water-sensors-to-monitor-water-quality-in-rivers-lakes-and-the-sea/. [Дата обращения: 09 04 2021].
7. «Консалтинговая компания "Верное решение",» [В Интернете]. Available: https://xn----dtbhaacat8bfloi8h.xn--p1ai/iot-industrial-water. [Дата обращения: 04 09 2021].

POSSIBILITY OF USING SENSORS FOR AUTOMATIC MONITORING OF THE KIZELOVSKY COAL BASIN

Nikitina Elena Yurievna, Svirepova Anna Andreevna

Perm State University, 15, Bukireva st., Perm, 614990, Russia, ann.svirepova@yandex.ru

Abstract. The problem of wastewater pollution in the Kizelovsky coal basin is considered. At the moment, the analysis of KUB wastewater is carried out by specialists manually and is very rare. To automate the process of collecting wastewater indicators, it is necessary to select suitable sensors. The classification of existing sensors is given. A selection of the type of sensors was made that are suitable for automating the process of collecting indicators of wastewater from the Kizelovsky coal basin. Water quality monitoring sensors of such companies as Libelium and Systemelektronik are considered. The comparative characteristics of the proposed sensors are given. The comparison was made according to four criteria: parameters that can be measured by sensors, methods of transmitting data from sensors, scalability of the sensor system, and the ability to remotely control sensors. Based on the compiled characteristics, a choice was made in favor of one of the considered sensor systems.

Key words: Kizelovsky coal basin, automation of the process of collecting indicators, water quality monitoring sensors.

УДК 004.413

ОПРЕДЕЛЕНИЕ ПОДХОДА К ИНТЕГРАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В ИНФОРМАЦИОННЫЕ СИСТЕМЫ ОПЕРАТОРОВ УСЛУГ СВЯЗИ

Радыгин Сергей Викторович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, Edge13@mail.ru

В статье ставится задача рассмотреть подходы интеграции программного обеспечения в информационные системы и оценить их с точки зрения эффективности и безопасности, а так же удобства использования. Возрастающая с каждым разом сложность разработки и обслуживания программного обеспечения, влечет за собой возникновение новых рисков и проблем. Обосновывается мысль о том, что человеческий фактор выступает главным риском при развертывании программного обеспечения, что в свою очередь может привести к серьезным проблемам безопасности, таким как раскрытие конфиденциальной информации о пользователях или же, о самой компании. В статье описываются подходы интеграции с возможности их автоматизации и уменьшения участия в них человека. Автор приходит к выводу, что автоматизация процесса развертывания устраняет подобные риски и дает возможность компании работать в гибком режиме, что позволяет повысить ценность бизнеса.

Ключевые слова: автоматизация, разработка, непрерывная интеграция, развертывание, программного обеспечения, риски.

Введение

Наиболее важная проблема, с которой сталкиваются разработчики в крупных компаниях, предоставляющие телекоммуникационные услуги, состоит в следующем: как в кратчайший срок предоставить новый функционал приложения после добавления в него нового кода? Новые полезные идеи необходимы для конкурентной борьбы за клиентов. Кроме того, после обнаружения ошибки в приложении необходимо как можно быстрее устранить ее и обеспечить непрерывную работу внутренних систем компании, предотвратив прерывание услуг для клиентов.

Обычно этап развёртывания приложения незаслуженно считают второстепенным этапом, хотя без хорошо отлаженной, надежной технологии развертывания новый функционал не может быстро и эффективно начать работу. Поэтому стоит сосредоточить внимание на процессах сборки, установки, тестирования и развертывания программного обеспечения.

Существует множество методов разработки программного обеспечения. В настоящее время их обычно делят на «гибкий» и «негибкий» подходы. Традиционный, негибкий подход к созданию программного обеспечения обычно опирается на более регламентированный процесс разработки. В качестве примера можно привести процесс типа «водопад», в котором последовательно выполняются любые действия по определению требований, проектированию, разработке и тестированию. Несмотря на то, что разработка указанным методом длительное время служила стандартом разработки больших и сложных систем, в ней есть несколько явных недостатков. К представителям такого подхода относится ручное развертывание. Кроме того, в случаях откладывания тестирования и интеграции до завершающих этапов проекта, проблемы могут зачастую выявляются слишком поздно, и их решение срывает необходимые сроки. Представителем такого подхода является

развертывание после завершения этапа разработки. Рассмотрим более подробно данные подходы.

Ручное Развертывание

Большинство современных приложений сложно развертывать вследствие того, что в них много взаимозависимых изменяющихся частей. Многие организации развертывают программное обеспечение вручную. Это означает, что процесс развертывания разбивается на ряд атомарных стадий, каждая из которых выполняется ответственным исполнителем или командой. На каждой стадии нужно принимать сложные решения, в результате чего повышается вероятность ошибки. И даже если не будет ошибок, различия в принципах управления разными стадиями приводят к нестабильным результатам. Эти различия могут повлиять на процесс развертывания негативным образом[1].

Можно выделить основные признаки развертывания программного обеспечения вручную:

1. В первую очередь это то, что приходится прилагать много усилий на составление документов перед началом проектирования, а также на проектирование перед началом программирования, хотя ясно, что требования со временем будут меняться. Создание большого объема документации, в которой необходимо подробно описать все шаги внесения изменения и разбор типичных ошибок – сложная задача. На ее решение тратится много времени, и требуется вовлечение многих людей. Тем не менее, документация чаще всего неполная или устаревшая.
2. Решение о работоспособности релиза принимается исключительно на основе результатов ручного тестирования. Что во многом опирается на человеческий фактор и может приводить к многочисленным мелким недочетам. Даже если тесты составлены хорошо, бывает тяжело отследить источники ошибок.
3. Частые обращения к команде разработчиков для получения объяснений, почему в день поставки релиза приложение работает не так, как предполагается.
4. Частые изменения и пробные запуски релизов занимают много времени.
5. Результаты установки релизов непредсказуемые. Часто нужно возвращаться назад и разбираться с возникающими проблемами.
6. Развертывание вручную весьма рутинный процесс, тем не менее, требующий высокой квалификации. Когда квалифицированные специалисты делают рутинную (хотя и сложную) работу, они неизбежно совершают ошибки самых разных типов.

Иногда ручное развертывание проходит довольно гладко, но чаще всего это не так. Общепризнанно, что процесс развертывания может содержать ошибки и существенно задерживать выпуск.

Из выше перечисленного следует, что применение ручного подхода требует большого количества времени и высокую квалификацию, хотя и это не гарантирует хорошего результата.

Развертывание после завершения этапа разработки

В этом сценарии приложение впервые развертывается в среде производственного типа только после того, когда команда разработчиков отчитается, что они уже почти все сделали и можно попробовать установить приложение.

Если в процессе разработки программного обеспечения участвовала команда тестирования, то изменения уже были проверены в среде разработки. Если же они отсутствовали, то тестирование готового приложения займет большое количество времени и возможно выявит множество ошибок, исправление которых вновь потребует прохождение всего этапа разработки.

По завершению этапа разработки команда разработчиков собирает в единый комплект все инсталляторы, конфигурационные файлы, процедуры переноса баз данных и документацию, чтобы передать комплект команде, которая будет заниматься развертыванием. Естественно, весь комплект не тестировался.

В некоторых случаях члены команды имеют все необходимые знания, но часто в крупных организациях команда развертывания делится на несколько специализированных групп. Многие стадии еще не тестировались в отладочной среде, поэтому часто возникают ошибки, в документации пропущены важные стадии. В результате чего этап развертывания превращается в долгий и очень сложный процесс.

В процессе развертывания нередко обнаруживается, что ошибочные предположения заложены в проект системы. На компьютере разработчика приложение будет работать хорошо, однако на сервере могут возникнуть серьезные проблемы. Решение подобных проблем может занять много времени, причем развертывание нельзя считать успешным, пока все такие проблемы не будут решены.

Как результат, развертывание программного обеспечения только по окончанию этапа разработки, это долгий, трудоемкий процесс, результат которого неплохо предсказуем. Использование данного подхода так же занимает большое количество времени и сил и не является оптимальным.

Непрерывная интеграция

На сегодняшний день многие крупные компании используют методы гибкой разработки. Они с выгодой для себя освоили эти итеративные и инкрементные методы коллективной разработки решений. При применении в рамках архитектурного подхода методы непрерывной интеграции и разработки через тестирование расширяют базовые методы гибкой разработки настолько, чтобы обеспечить высокое качество и гибкость проектов.

Внедрение такой системы в проекты компании будет являться ещё одним элементом гибкой методологии разработки, что позволит проектам компании развиваться более динамично в быстро меняющихся условиях рынка, а также поставлять более качественный продукт конечным пользователям.

Для начала дадим определение непрерывной интеграции, а затем более подробно рассмотрим этот подход к развертыванию программного обеспечения.

Непрерывная интеграция – практика разработки программного обеспечения, которая заключается в постоянном слиянии рабочих копий в общую основную ветвь разработки (до нескольких раз в день) и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем[2].

Главное преимущество такого подхода состоит в том, что он позволяет превратить выпуск новых версий в повторяемый, надежный и предсказуемый процесс. В результате, существенно уменьшается продолжительность цикла поставки, т.е. интервал времени между принятием решения об изменении (устранении ошибки или добавлении нового функционала) и моментом, когда релиз становится доступным для пользователей.

Экономия денежных затрат, обусловленная сокращением продолжительности цикла, удешевлением поддержки продукта и уменьшением количества ошибок, перекрывает все затраты на реализацию системы непрерывного развертывания.

Один из ключевых принципов непрерывного развертывания состоит в том, что он, является системой обслуживания по запросу и позволяет командам тестирования, администраторам и техперсоналу свободно работать с очередной версией приложения в выбранной ими среде. Каждый видит, какие версии доступны для развертывания и может выполнить развертывание простым щелчком на кнопке.

Как только сборка готова и протестирована – она готова к развёртыванию. Ключевой идеей непрерывной интеграции является то, что если сборка и тестирование прошли успешно, то нет причин откладывать развёртывание данной сборки. Конечно, в реальном мире имеют место быть причины отложить развёртывание: например, новый функционал может быть развернут только поверх критического обновления, которое в свою очередь еще не прошло развертывание. Поэтому финальное решение за развёртыванием сборки должно оставаться за человеком, и система должна предоставлять возможность держать сборку готовой к развёртыванию в любой момент[3].

В результате этого в каждый момент времени для отдельных команд и членов команд доступны несколько разных версий в разных средах. Возможность легко развернуть любую версию в любой среде предоставляет ряд преимуществ.

Тестировщики могут вернуться к предыдущим версиям, чтобы проанализировать изменение поведения по сравнению с новыми версиями. Техперсонал может развернуть релиз для воспроизведения дефектов. Системный администратор может выбрать приемлемую версию сборки для развертывания в рабочей среде в ходе восстановления системы после возникновения критических ошибок.

Гибкость, предоставляемая инструментами развертывания, повышает качество работы. В целом это приводит к тому, что члены команды лучше контролируют свою работу, и качество программного продукта улучшается. Команды более эффективно взаимодействуют друг с другом, уменьшается количество конфликтов, и не приходится ждать появления стабильной версии.

Можно выделить следующие преимущества непрерывной интеграции и автоматизации развертывания программного обеспечения.

1. Развертывание осуществляется чаще – вплоть до нескольких раз в день. Это сокращает время ввода в действие новых возможностей.
2. Частые развертывания также ускоряют получение отзывов на новые особенности и изменения в коде. Разработчикам не приходится вспоминать, что делалось в прошлом месяце.
3. Автоматизация дает больше гибкости. Тестовые окружения можно создавать по мере необходимости. Например, в случае изменения пользовательского интерфейса можно создать отдельное тестовое окружение для рекламы на ограниченный период, а для всестороннего нагрузочного тестирования отдельные окружения, близкие по своим параметрам к рабочему, и уничтожить их после тестирования, чтобы не вкладывать деньги в ненужные аппаратные средства.
4. Кроме того, автоматизация тестирования упрощает воспроизведение ошибок. Поскольку во время каждого теста выполняется одна и та же последовательность действий, с процедурой выполнения теста не связано никаких ошибок. Автоматизированные тесты могут выполняться чаще и не требуют дополнительных усилий.
5. Риски, связанные с установкой новых версий, существенно уменьшаются за счет такой настройки процедуры развертывания в рабочем окружении, которая дает возможность легко откатиться к старой версии, если это потребуется. Это позволяет предотвратить простои действующего сервера.
6. Наконец, приложения, как предполагается, находятся под постоянным «присмотром» – мониторингом, поэтому неожиданная остановка любого процесса, например отвечающего за регистрацию, не останется незамеченной.

Заключение

Подводя итоги, можно сказать, что непрерывное развертывание позволит бизнесу быстрее внедрять новые возможности и увеличит надежность компьютерных систем. Повышенную надежность можно рассматривать как дополнительный ресурс для разработчиков. Кроме того, гораздо лучше, если разработчики и сотрудники предприятия обнаруживают ошибки во время тестирования, а не после развертывания нового кода в рабочем окружении [4].

Возрастающая с каждым разом сложность разработки и обслуживания программного обеспечения вызывает все больший интерес и внимание со стороны программистов и компаний, особенно предоставляющих какие-либо услуги конечным пользователям, в том числе операторов услуг связи.

Цель технологии непрерывного развертывания – создать быстрый и надежный процесс переноса программного обеспечения в рабочее окружение и в кратчайший срок

предоставить пользователям очередную версию приложения после добавления в него нового функционала.

Библиографический список

1. *Поль М. Дюваль, Стивен М. Матиас III, Эндрю Гловер*. Непрерывная интеграция: улучшение качества программного обеспечения и снижение риска. Вильямс, 2008.
2. *Википедия*. Свободная энциклопедия. Непрерывная интеграция [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Непрерывная_интеграция (дата обращения: 19.04.2021).
3. *Хамбл Джез, Фарли Дэвид*. Непрерывное развертывание ПО. Автоматизация процессов сборки, тестирования и внедрения новых версий программ. Вильямс, 2016.
4. *Вольф Э*. Continuous delivery. Практика непрерывных апдейтов / Э. Вольф – Питер СПб, 2018.

DEFINING THE INTEGRATION APPROACH INTO THE INFORMATION SYSTEMS OF COMMUNICATION SERVICE OPERATORS

Radygin Sergey V.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, Edge13@mail.ru

The article aims to consider approaches to software integration in information systems and evaluate them from the point of view of efficiency and security, as well as usability. Increasing complexity of software development and maintenance entails new risks and problems. It substantiates the idea that the human factor is the main risk in software deployment, which in turn can lead to serious security problems, such as disclosure of confidential information about users or the company itself. The article describes integration approaches with the possibility of automating them and reducing human involvement in them. The author concludes that automating the deployment process eliminates such risks and allows the company to operate in a flexible mode, thus increasing business value.

Keywords: automation; development; continuous integration; deployment; software; risks.

УДК 519-6, 004.421

МОДЕЛЬ СТРУКТУРЫ ТОНКОЙ ПЛЕНКИ ЭПОКСИДНОЙ СМОЛЫ, МОДИФИЦИРОВАННОЙ УГЛЕРОДНЫМИ НАНОТРУБКАМИ, С УЧЕТОМ НАЛИЧИЯ ВАН-ДЕР-ВААЛЬСОВА ВЗАИМОДЕЙСТВИЯ

Романова Марина Павловна, Бузмакова Мария Михайловна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, romanovamp@yandex.ru

В настоящей работе предложена континуальная перколяционная модель структуры тонкой пленки эпоксидной смолы, модифицированной углеродными нанотрубками, учитывающей наличие Ван-дер-Ваальсова взаимодействия. Углеродные нанотрубки представлены *k*-мерами (отрезками длиной *k*), равномерно распределенными в полимерной матрице – двумерной плоскости, с наличием межфазного взаимодействия *k*-меров. При моделировании использованы периодические граничные условия. Для проведения численного эксперимента по моделированию структуры полимерного нанокомпозита с учетом дисперсионного взаимодействия разработаны эффективные алгоритмы равномерного распределения *k*-меров на плоскости, распределения *k*-меров по кластерам, поиска перколяционного кластера и порога перколяции, который соответствует критической концентрации *k*-меров, при которой наноматериал меняет свои свойства. Получены удовлетворительные результаты порогов перколяции для различных размерностей плоскостей *L* с учетом межфазных областей.

Ключевые слова: математическое и компьютерное моделирование, полимер, углеродные нанотрубки, теория перколяции.

Введение

При добавлении наночастиц в полимерный материал, его свойства могут быть значительно улучшены [1]. Одним из нанонаполнителей, наиболее привлекающим внимание исследователей, являются углеродные нанотрубки (УНТ), которые улучшают следующие характеристики полимера: электропроводность, теплопроводность, различные механические свойства и другие [2]. Поведение таких материалов в зависимости от концентрации, ориентационном упорядочении, а также межчастичном и межфазном взаимодействиях между частицами мало изучены. Создание полимеров, модифицированных углеродными нанотрубками, требует больших затрат временных и денежных ресурсов, поэтому возникает потребность в создании математической и компьютерной модели полимеров для теоретического исследования их структуры и свойств.

Постановка задачи

Предложена континуальная перколяционная модель структуры тонкой пленки полимера, модифицированного углеродными нанотрубками с учетом наличия Ван-дер-Ваальсова взаимодействия. Изучены свойства данной модели в зависимости от величины межфазного слоя. В рамках модели полимерная матрица представлена конечной двумерной системой, случайно заполненной линейными *k*-мерами – моделями углеродных нанотрубок, различной длины (*k* – среднее значение длины, отсюда название *k*-мер) с заданной шириной $d = 1$ – диаметром углеродной нанотрубки. Межфазные области учитываются путем добавления проницаемого слоя у *k*-меров, где минимальная толщина этого слоя характеризуется простой химической связью *C-N*. Ван-дер-Ваальсово взаимодействие учитывается путем добавления вероятности возникновения связи между *k*-мерами, при пересечении их проницаемых слоев.

Основным результатом является определение порога перколяции − концентрации упакованных *k*-меров, при которой вероятность возникновения перколяционного кластера равна 0.5. Перколяционный кластер – кластер, соединяющий две противоположные стороны системы. Порог перколяции соответствует критической концентрации углеродных нанотрубок в полимере, при которой наноматериал значительно меняет свои свойства.

Предложенная модель может быть описана следующим математическим соотношением

$$M = <L, d, p, R\{x_i, y_i, k_i, a_i\}, K, mF>,$$

где L – линейный размер квадрата, d – ширина k – мера, p – доля площади квадрата, занимаемая *k*-мерами, $R\{x_i, y_i, k_i, a_i\}$– множество координат начал отрезков и углов их ориентации, $k_i = a \pm \sigma^2$ – длина *k*-мера, задаваемая математическим ожиданием и среднеквадратичным отклонением, K – количество испытаний, mF – величина проницаемого слоя.

Методы моделирования

Моделирование структуры тонкой пленки полимера проводилось методами Монте-Карло [3]. Для генерации псевдослучайных чисел использован алгоритм «Вихрь Мерсенна», который обеспечивает быструю генерацию высококачественных по критерию случайности чисел [4]. Углы и координаты начала *k*-меров подчиняются равномерному закону распределения, а длины *k*-меров − нормальному закону распределения. Для достижения нормального распределения значений длин *k*-меров используется преобразование Бокса-Мюллера [5]. Для исключения влияния поверхности при моделировании использовались периодические граничные условия.

Для удовлетворения условия случайности возникновения Ван-дер-Ваальсова взаимодействия введен параметр $G(0;1)$ – сгенерированная случайная величина, удовлетворяющая равномерному распределению, на интервале (0;1). Тогда при пересечении проницаемых слоев *k* -меров и при условии $\frac{U_{real}}{U_{max}} \leq G(0;1)$ фиксируется возникновение связи

между такими *k*-мерами, и они относятся к одному кластеру, где $U_{real} \sim \frac{\alpha_{||}^2}{D_{real}^6}$, $U_{max} \sim \frac{\alpha_{||}^2}{D_{max}^6}$, D – кратчайшее расстояние между *k*-мерами, $\alpha_{||} = (0.23 + 0.135d)(k^2 + 52.5)$ – величина продольной поляризуемости УНТ [6].

Данные вычислительного эксперимента аппроксимируются функцией

$$P(p) = (1 + \exp(-(p - p_c(L))a))^{-1}.$$

При аппроксимации экспериментальных данных определяется порог перколяции для квадратной плоскости конечного размера *L*. Методика определения порога перколяции подробно описана в [7]. Для экстраполяции значений проницаемого слоя для бесконечных систем используется скейлинговое отношение:

$$|p_c(L) - p_c(\infty)| \infty L^{-1/\nu},$$

где ν – универсальный критический показатель и равен 4/3 в случае двумерных перколяционных задач.

Результаты и обсуждение

Для данной модели были получены результаты численного эксперимента для набора следующих входных параметров: линейный размер квадрата L = 1000, 2000, 3000; длины k-меров, подчиняющиеся нормальному закону распределения, k = 100±10 (100 – среднее значение аспектного отношения – длины к диаметру малой одностенной углеродной нанотрубки); количество испытаний K = 100. Значения параметров L и k задаются в диаметрах d. В таблице 1 представлены результаты порогов перколяции в зависимости от различных размерностей квадрата *L* и величины проницаемого слоя *mF*, а также порогов перколяции для бесконечных систем.

Таблица 1. Значение порогов перколяции

		L			
		$p_c(\infty)$	1000	2000	3000
mF	2,5	0,0694 ± 9,2359E-5	0,0683 ± 3,1173E-4	0,0688 ± 4,3423E-4	0,0689 ± 3,9899E-4
	5	0,0352 ± 9,2389E-5	0,0331 ± 8,8092E-5	0,0339 ± 8,4262E-5	0,0343 ± 8,8164E-5
	7,5	0,0267 ± 1,4626E-4	0,0255 ± 7,5747E-5	0,0259 ± 6,1243E-5	0,0262 ± 4,3119E-5
	10	0,0204 ± 4,6194E-5	0,0192 ± 6,4265E-5	0,0197 ± 2,4548E-5	0,0199 ± 1,3668E-4
	12,5	0,0175 ± 3,6179E-4	0,0168 ± 5,2282E-5	0,0169 ± 2,8039E-5	0,0173 ± 4,5212E-5
	15	0,0147 ± 5,3873E-5	0,0138 ± 3,8819E-5	0,0141 ± 3,8366E-5	0,0143 ± 9,7589E-5
	17,5	0,0133 ± 4,2337E-4	0,0122 ± 4,8610E-5	0,0126 ± 1,6695E-5	0,0130 ± 1,1999E-5
	20	0,0110 ± 1,7704E-4	0,0107 ± 3,6886E-5	0,0109 ± 3,412E-5	0,0108 ± 3,0555E-5

Результаты показывают снижение значения порогов перколяции при увеличении параметра *mF*.

Авторами настоящей работы была создана и исследована континуальная перколяционная модель структуры тонкой пленки полимера, модифицированного УНТ, учитывающую наличие Ван-дер-Ваальсова взаимодействия и межфазных областей. Для модели разработана программа для проведения численного эксперимента по определению порога перколяции, который соответствует значению критической концентрации углеродных нанотрубок в тонкой пленке полимера, при которой нанокомпозит значительно меняет свойства.

Была получена зависимость порога перколяции от ряда значений межфазного слоя. Замечено, что для всех размерностей системы зависимость порога перколяции от межфазного слоя является нелинейной. В дальнейшем планируется более подробное исследование и анализ полученных результатов, получение аппроксимирующих функций, а также изучение поведения порога перколяции при изменении среднеквадратичного отклонения длин углеродных нанотрубок.

Библиографический список

1. *Ю.Ю. Тарасевич.* Перколяция: теория, приложения, алгоритмы: Учебное пособие. М.: Елиториал УРСС. 2002. С.112.
2. *А.В. Елецкий, А.А. Книжник, Б.В. Потапкин, Х.М. Кенни.* Электрические характеристики полимерных композитов, содержащих углеродные нанотрубки // Успехи физических наук. 2015. В.185, № 3. С.225 – 270.
3. *Metropolis, N., Ulam, S.* The Monte Carlo Method // Journal of the American Statistical Association. 1949. V.44, №247. P. 335-341.
4. *Matsumoto M., Nishimura T.* Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator» // ACM Trans. on Modeling and Computer Simulations. 1998. V.8, №1. P. 3-30.
5. *Гельгор А. Л.* Методы моделирования случайных величин и случайных процессов. СПб.: Издательство политехнического университета. 2012. С. 217.
6. *Г.С. Бочаров, А.А. Книжник, А.В. Елецкий, T.J. Sommerer* // Влияние электрического поля на ориентацию углеродных нанотрубок в процессе их роста и эмиссии. // Журнал технической физики. 2012. Т.82, В.2. С.113-121.
7. *Бузмакова М. М.* Компьютерное моделирование континуальной перколяции сфер и эллипсоидов с проницаемыми оболочками // Диссертация на соискание ученой степени кандидата физико-математических наук 05.13.18 – математическое моделирование, численные методы и комплексы программ, Астрахань, 2013. С.168.

MODEL OF THE STRUCTURE OF A THIN FILM OF EPOXY RESIN MODIFIED BY CARBON NANOTUBES, TAKING INTO ACCOUNT THE PRESENCE OF THE VAN DER WAALS INTERACTION

Romanova Marina P., Buzmakova Mariya M.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, romanovamp@yandex.ru

The continuum percolation model of the structure of epoxy thin film, modified by carbon nanotubes is proposed in the paper. The model takes into account the presence of the van der Waals interaction and interphase regions. The model includes a L-side square polymer matrix, k-length k-meres carbon nanotubes and interphase regions of k-meres soft shells. For the model effective algorithms was developed for the uniform k-meres distribution on the square, for the clustering of k-meres, for the search of percolation cluster and for detect of percolation threshold. The percolation threshold corresponds to the critical concentration value of carbon nanotubes in polymer (in which nanocomposite change properties). The computer program was created. The computer modeling

was carried out using Monte Carlo Methods. The correlation between percolation threshold and k-meres interphase regions was found.

Key words: mathematical and computer modeling, polymer, carbon nanotubes, percolation theory.

УДК 004.056.53

ИССЛЕДОВАНИЕ ПАРАМЕТРОВ ЭЛЛИПТИЧЕСКИХ КРИВЫХ В ФОРМЕ СКРУЧЕННЫХ ЭДВАРДСА

Семыкина Екатерина Дмитриевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, semykina.ek@gmail.com

В данной работе исследуется проблема подбора параметров для эллиптической кривой в форме скрученных кривых Эдвардса. Проводится краткий теоретический обзор эллиптических кривых, представленных в форме Вейерштрасса и скрученных кривых Эдвардса, рассматривается их применение в криптографии, затрагиваются принципы формирования и проверки электронной подписи на эллиптических кривых. Обосновывается актуальность использования эллиптических кривых в форме скрученных Эдвардса для ГОСТ группы 34.10. Написана программа, генерирующая для заданного модуля подходящие кривые. Для исследования сгенерировано более 300 000 различных кривых. Полученные данные подвергаются анализу. Осуществляется поиск зависимостей параметров скрученной эллиптической кривой Эдвардса. На основе полученных результатов созданы рекомендации по выбору значений параметров эллиптической кривой.

Ключевые слова: эллиптические кривые, скрученные Эдвардса, ГОСТ 34.10-2018, параметры эллиптических кривых.

Введение

В современном мире криптография играет одну из ключевых ролей среди существующих методов обеспечения информационной безопасности. Сегодня криптосистемы на эллиптических кривых используются в важнейших технологиях, на которых базируются современный информационный мир. Существует множество форм представления кривых, особую роль занимает кривая в форме скрученной Эдвардса. Данные кривые имеют большие перспективы в криптографии, поскольку обладают рядом полезных свойств.

Целью данной работы является исследование зависимостей параметров скрученных кривых Эдвардса, на основе которых могут быть даны рекомендации по выбору подходящих вариантов значений параметров для группы ГОСТ 34.10.

Эллиптические кривые.

Эллиптическая кривая в форме Вейерштрасса является каноническим представлением, в приложении к криптографии – это множество пар *(x, y)*, удовлетворяющих уравнению:

$$y^2 = x^3 + ax + b \ (mod\ p)$$

где a, b принадлежат полю F(p).

Скрученные кривые Эдвардса описываются следующим уравнением [1]:

$$ax^2 + y^2 = (1 + dx^2y^2), \quad a, d \in F_p^*$$

В 2014 году в своем докладе [2] сотрудники компании КриптоПро наглядно продемонстрировали возможности преобразования различных форм представления эллиптических кривых. Уже в 2016 году на основании работ КриптоПро Федеральным агентством по техническому регулированию и метрологии были выпущены рекомендации по стандартизации «Криптографическая защита информации. Параметры эллиптических кривых для криптографических алгоритмов и протоколов» [3]. В этом документе описаны математические преобразования для указанных выше форм кривых.

Применение в криптографии

В данной работе большое внимание уделяется группе ГОСТ 34.10 – это стандарты, описывающие алгоритмы формирования и проверки электронной цифровой подписи, основанные на эллиптических кривых. Стойкость алгоритма основывается на сложности вычисления дискретного логарифма в группе точек эллиптической кривой, а также на стойкости хэш-функции. Особенностью данной группы стандартов является тот факт, что в документах не зафиксированы какие-либо кривые, рекомендуемые для использования, уточняется лишь набор ограничений к ним.

Преимущества использования кривых в форме скрученных Эдвардса

Согласно ГОСТ группы 34.10, в канонической форме (форме Вейерштрасса) необходимо представлять результаты вычислений, при этом ограничений на форму, в которой производятся промежуточные вычислений, стандарт не устанавливает. Можно выделить следующие основные преимущества использования кривых в форме скрученных Эдвардса в сравнении с кривыми Вейерштрасса [2]

1) возможность построения более эффективных реализаций

По результатам ряда исследований компании КриптоПро [2] стало известно, что закон сложения точек на скрученных кривых в форме Эдвардса допускает построение реализаций стандартов группы ГОСТ 34.10, позволяющие увеличить скорость вычислений. Согласно исследованиям, средняя доля ускорения по трем основным операциям (формирование подписи, проверка подписи и выработка ключа) для кривых с модулем размера 256 бит составило 20%, а для кривых с модулем размера 512 битов – 25%.

2) однородность закона сложения

Известно, что для кривых в форме Вейерштрасса алгоритм сложения точек отличается для разных точек и одинаковых точек. Различие алгоритмов, в свою очередь, влияет на скорость данной криптографической операции, что может привести к возможности реализации атаки по побочным каналам. Для скрученных Эдвардса закон сложения точек формулируется одним алгоритмом, что практически исключает угрозу данной атаки.

Исследование параметров эллиптических кривых в форме скрученных Эдвардса

Входным параметром разработанного алгоритма является простое число p, используемое в качестве модуля эллиптической кривой. Для данного входного p генерируются эллиптические кривые в форме скрученных Эдвардса с параметрами $0 < e < p$ и $0 < d < p$.

С помощью преобразований, описанных в [3], каждая эллиптическая кривая приводится к канонической форме, после чего кривая в форме Вейерштрасса проверяется на ограничения ГОСТ 34.10. При этом кривая включается в список, только если ее порядок равен $N = 4*q$, где q – простое число. В ходе работы программы было сгенерировано 327452 кривых.

Для выявления зависимостей параметров p, e, d, было принято решение исследовать «плотность» покрытия диапазона всех значений параметров группой параметров, для которых кривая существует и подходит для ГОСТа, чтобы на основе обнаруженных зависимостей можно было выявить статистические критерии выбора кривой.

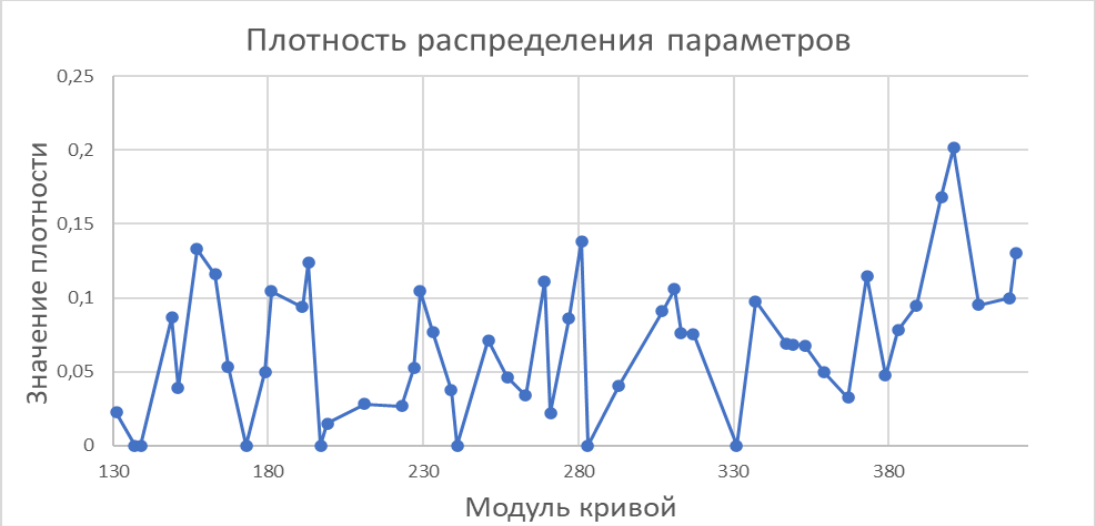


Рис. 1 Плотность распределения параметров

По графику (рис.1) видим, что плотность покрытия подходящими кривыми всех возможных сочетаний параметров принимает значения хаотично, ярко выраженной зависимости не наблюдается. Есть как спады до нулевого значения (при модуле кривых, для которых подходящих кривых не было обнаружено), так и пики. Так, например, из всех возможных сочетаний параметров для фиксированного модуля кривой $p = 401$ к использованию пригодны около 20%, а при фиксированном модуле $p = 331$ подходящие кривые не были обнаружены совсем.

На втором шаге изучим непосредственно значения параметров, при которых существует необходимая кривая. Построим распределения для значений параметров e и d соответственно.

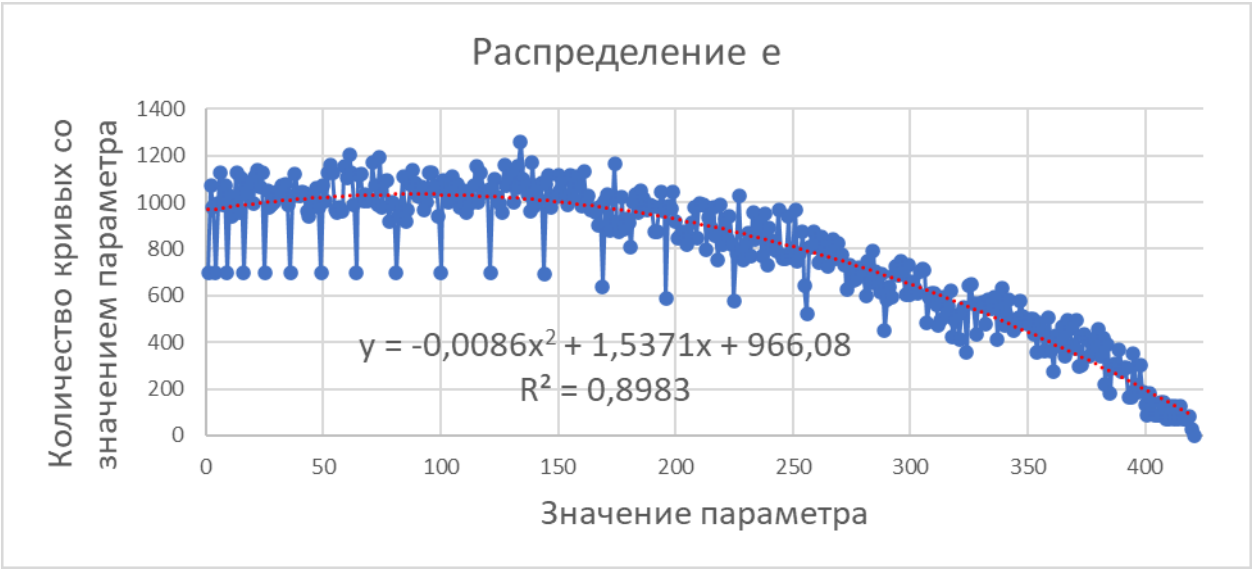


Рис. 2 Распределение значений параметра e

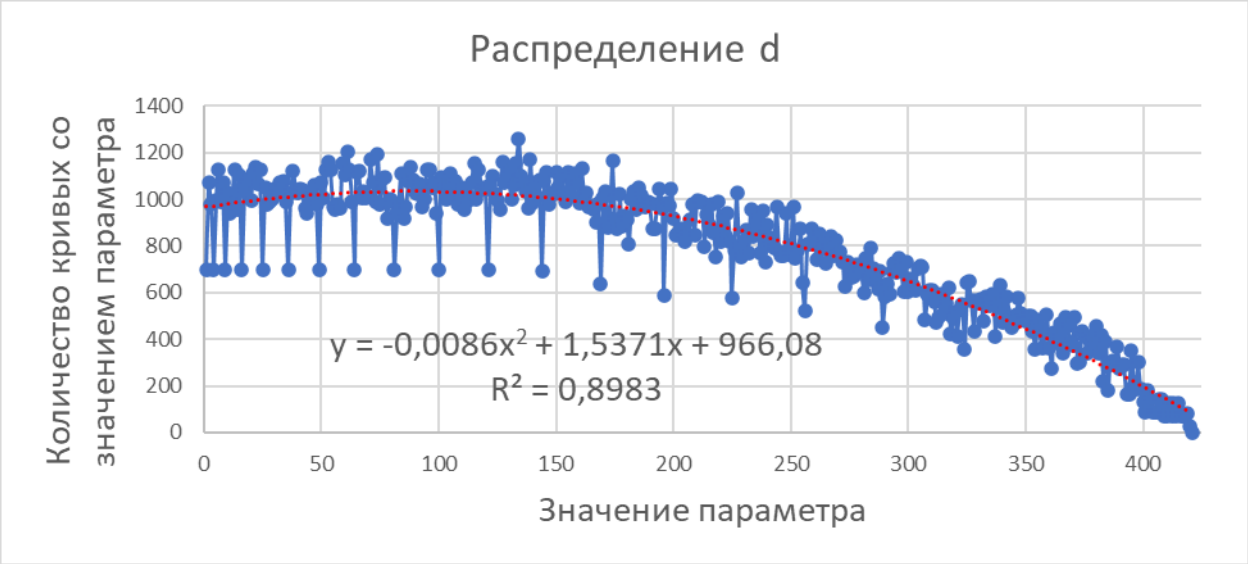


Рис. 3 Распределение значений параметра d

По построенным графикам (рис. 3, 4) видим, что распределения значений параметров *e, d* для всего объема сгенерированных кривых получились идентичными.

Наиболее удачной в нашем случае оказалась аппроксимация к полиномиальной функции (степень 2)

y = -0,0086x^2 + 1,5371x + 966,08

Также на данных графиках (рис. 3,4) распределения параметров мы можем наблюдать точечные спады с увеличивающимся интервалом. В ходе исследования выяснилось, что значения параметров, для которых наблюдается данная тенденция, равны квадратам натуральных чисел. При этом для распределения параметров *a, b* изоморфных кривых в форме Вейерштрасса подобной особенности обнаружено не было.

Далее проанализируем зависимость параметров друг от друга и от модуля кривой. Для этого построим распределения разностей значений параметров.

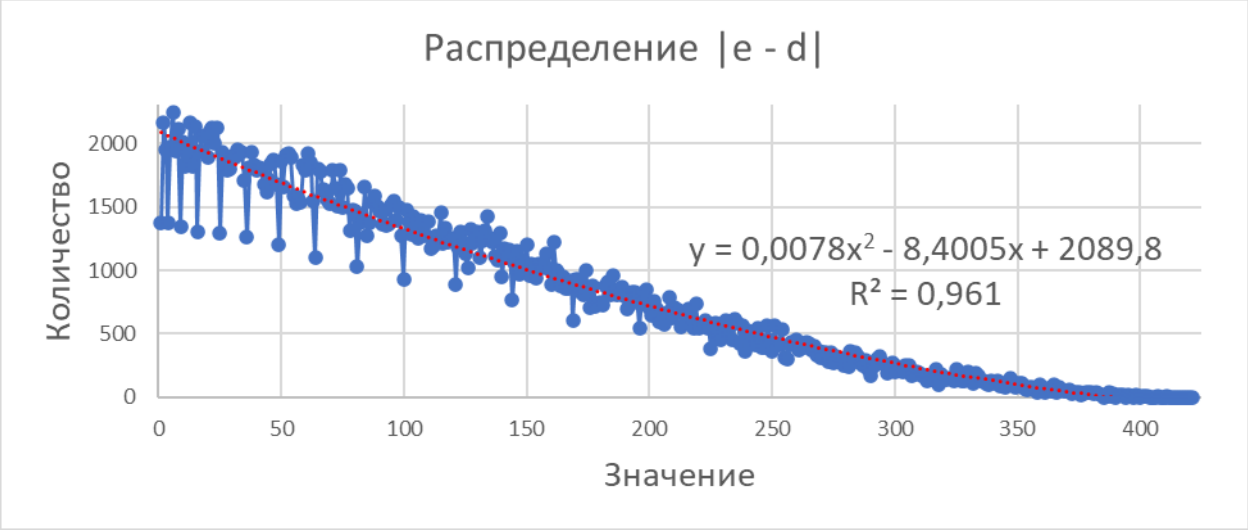


Рис. 4 Распределение значений разности параметров e, d

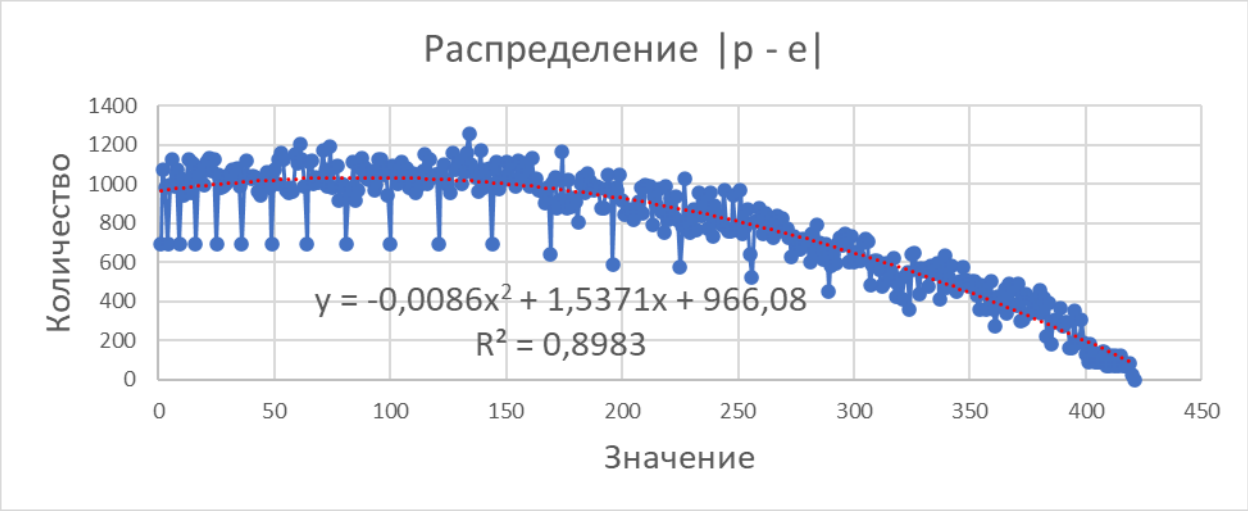


Рис. 5 Распределение значений разности параметров p, e

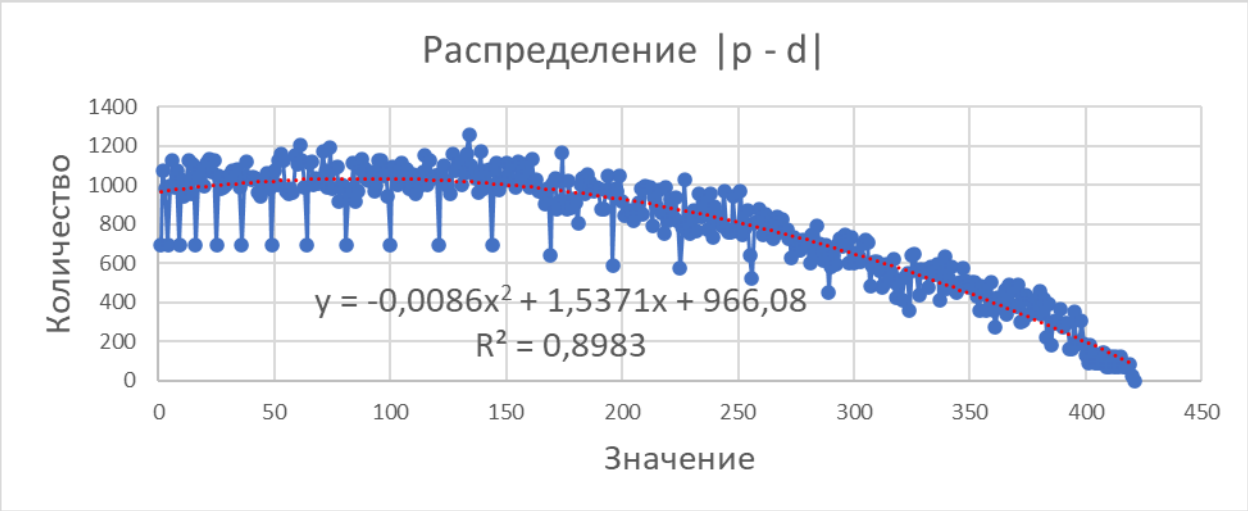


Рис. 6 Распределение значений разности параметров p, d

Для данных графиков распределений (рис 4, 5, 6) характерна все та же тенденция, наблюдаемая ранее: спады количества происходят при значениях разницы, равной квадрату натуральных чисел (1, 4, 9, 16, и т.д.).

Заключение

В ходе выполнения данной исследовательской курсовой работы были изучены математические основы различных форм эллиптических кривых, в частности эллиптических кривых в форме скрученных Эдвардса и Вейерштрасса, их применение в криптографии. Также были изучены ограничения, накладываемые на параметры эллиптических кривых современными стандартами и существующие рекомендации по их выбору. Целью настоящей курсовой работы являлось нахождение зависимостей параметров эллиптической кривой Эдвардса и формирование рекомендаций по генерации данных эллиптических кривых.

Цель, поставленная в начале исследования, достигнута. Составлены следующие рекомендации по выбору параметров:

1. Параметры *e, d* следует выбирать таким образом, чтобы их значения не были равны квадрату натурального числа.
2. Параметры *e, d* следует выбирать таким образом, чтобы разница их значений ($|e - d|$) не была равна квадрату натурального числа.
3. Параметры *e, d* следует выбирать таким образом, чтобы разница их значений с модулем кривой ($p - e$, $p - d$), также не была равна квадрату натурального числа.

Библиографический список

1. *Бессалов А.В.* Эллиптические кривые в форме Эдвардса и криптография: монография. – Киев: ІВЦ «Видавництво «Політехника»», 2017. –272с.
2. *Алексеев Е.К. Ошкин И.Б., Попов В.О., Смышляев С.В., Сонина Л.А.* О перспективах использования скрученных эллиптических кривых Эдвардса со стандартом ГОСТ Р 34.10-2012 и алгоритмом ключевого обмена на его основе. [Электронный ресурс]. – Режим доступа: https://www.ruscrypto.ru/resource/archive/rc2014/files/03_alekseev.pdf, свободный. – Загл. с экрана
3. *Рекомендации* по стандартизации Р 1323565.1.024-2019. Информационная технология. Криптографическая защита информации. Параметры эллиптических кривых для криптографических алгоритмов и протоколов. – Взамен Р 50.1.114– 2016; утверждены и введены в действие Приказом Федерального агентства по техническому регулированию и метрологии от 15 мая 2019 г. № 190-ст. – М. : Стандартинформ, 2019. – 11 с.

STUDY OF ELLIPTIC CURVE PARAMETERS IN THE FORM OF TWISTED EDWARDS

Semykina Ekaterina D.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, semykina.ek@gmail.com

This paper explores the problem of selecting parameters for an elliptic curve in the form of twisted Edwards. A short theoretical review of elliptic curves presented in the form of Weierstrass and twisted Edwards is carried out, their application in cryptography is considered, the principles of formation and verification of electronic signature on elliptic curves are touched upon. The relevance of using elliptic curves in the form of twisted Edwards for GOST group 34.10 is justified. A program is written that generates suitable curves for a given module. More than 300,000 different curves were generated for the study. The obtained data are analyzed. The parameter constraints of the twisted elliptical Edwards curve are searched. Based on the results, recommendations have been made for selecting elliptical curve parameter values.

Keywords: elliptical curves, twisted Edwards, GOST 34.10-2018, parameters of elliptic curves.

УДК 681.513.3

ПРИНЦИП РАБОТЫ СИСТЕМЫ МОНИТОРИНГА ZABBIX

Санников Сергей Николаевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, serega.sann@mail.ru

Аннотация: исследование посвящено изучению возможностей системы мониторинга Zabbix и выявлению перспективы ее модификации для использования в системах связи на основе первичных мультиплексоров. В частности, значительное внимание было уделено SNMP составляющей Zabbix и связанным с ней MIB (Management Information Base), которые предназначены для хранения и обработки информации об авариях. В результате было написано 12 MIB для обработки информации о авариях, таких как перегрев плат, отсутствие сигнала от устройства, высокое/низкое входное напряжение устройства и других.

Ключевые слова: система управления, система мониторинга, SNMP-агент, SNMP-менеджер.

Системы мониторинга используются в области связи повсеместно. Функция мониторинга предназначена для тех, кто хочет получить мониторинг инфраструктуры высоких уровней. В большинстве случаев управляющие IT-инфраструктуры не заинтересованы в низкоуровневых деталях, таких как недостаток места на диске, высокая загрузка процессора и т.д. Они заинтересованы в доступности сервиса, предоставляемым их IT-отделом. Управляющие также могут быть заинтересованы в выявлении слабых мест в IT инфраструктуре, SLA (Service Level Agreement – Соглашение об уровне услуг) различных IT услуг, структуре существующей IT-инфраструктуры, и в другой информации на более высоком уровне. Целью данного исследования является изучение SNMP-составляющей Zabbix и оценка перспективы модификации данной системы для работы с сетями на основе первичных мультиплексоров.

1. SNMP

SNMP (Simple Network Management Protocol) – сетевой протокол управления устройствами, способными общаться по IP протоколу в сетях на основе TCP/UDP.

Данный протокол был разработан с целью проверки функционирования сетевых маршрутизаторов и мостов. Впоследствии сфера действия протокола охватила и другие сетевые устройства, такие как хабы, шлюзы, терминальные сервера, LAN Manager сервера, машины под управлением Windows NT и т.д. Кроме того, протокол допускает возможность внесения изменений в функционирование указанных устройств. Именно посредством него выполняется взаимодействие между компонентами в данной работе.

2. Компоненты SNMP. Модель агент-менеджер

Основными взаимодействующими лицами протокола являются агенты и системы управления (менеджеры). Если рассматривать эти два понятия на языке "клиент-сервер", то роль сервера выполняют агенты, то есть те самые устройства, для опроса состояния которых и был разработан рассматриваемый нами протокол. Соответственно, роль клиентов отводится системам управления – сетевым приложениям, необходимым для сбора информации о функционировании агентов. Помимо этих двух субъектов в модели протокола можно выделить также еще два: управляющую информацию и сам протокол обмена данными.

Схема менеджер – агент позволяет строить достаточно сложные в структурном отношении распределенные системы управления.

Обычно, распределенная система управления включает большое количество связок менеджер – агент, которые дополняются рабочими станциями операторов сети, с помощью которых они получают доступ к менеджерам (рис. 1.). Каждый агент собирает данные и управляет определенным элементом сети. Менеджеры, иногда также называемые серверами системы управления, собирают данные от своих агентов, обобщают их и хранят в базе данных.

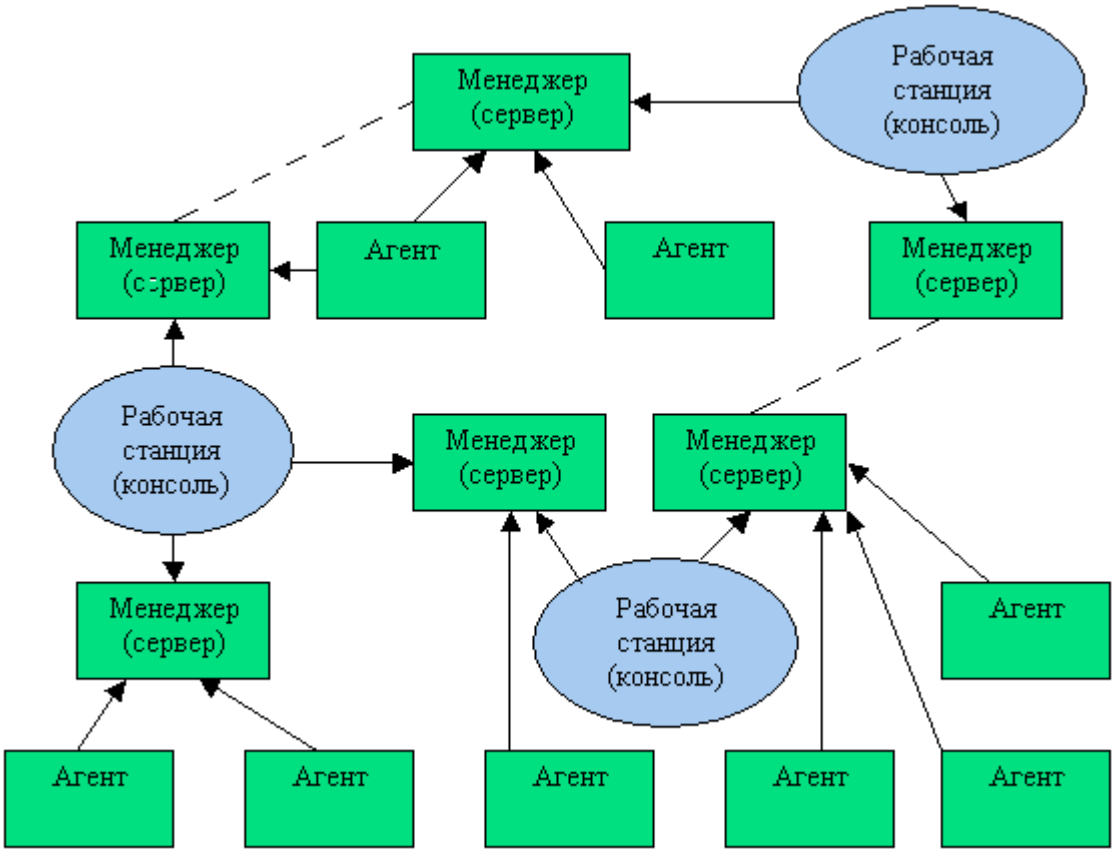


Рис.1 Схема менеджер-агент

Наличие нескольких менеджеров позволяет распределить между ними нагрузку по обработке данных управления, обеспечивая масштабируемость системы.

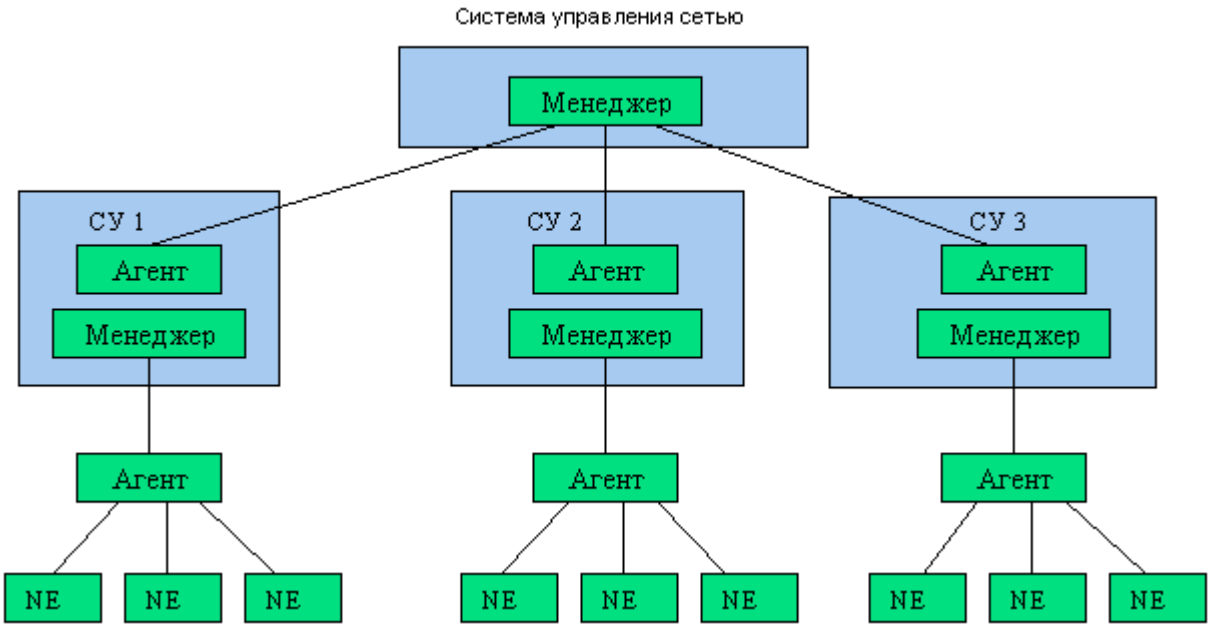


Рис.2 Иерархическая модель менеджер-агент

Гораздо более гибким является иерархическое построение связей между менеджерами (рис. 2). Каждый менеджер нижнего уровня выполняет также функции агента для менеджера верхнего уровня. Такой агент работает уже с гораздо более укрупненной моделью (MIB) своей части сети, в которой собирается именно та информация, которая нужна менеджеру верхнего уровня для управления сетью в целом. Обычно для разработки моделей сети на разных уровнях проектирование начинают с верхнего уровня, на котором определяется состав информации, требуемой от менеджеров-агентов более низкого уровня, поэтому такой подход назван подходом «сверху вниз». Он сокращает объемы информации, циркулирующей между уровнями системы управления, и приводит к гораздо более эффективной системе управления.

3. База управляющей информации (MIB)

Вся информация об объектах системы-агента содержится в так называемой MIB (management information base) – базе управляющей информации, другими словами, MIB представляет собой совокупность объектов, доступных для операций записи-чтения для каждого конкретного клиента, в зависимости от структуры и предназначения самого клиента. Не имеет смысла спрашивать у терминального сервера количество отброшенных пакетов, так как эти данные не имеют никакого отношения к его работе, так как и информация об администраторе для маршрутизатора. Потому управляющая система должна точно представлять себе, что и у кого запрашивать.

Для именования переменных базы MIB и однозначного определения их форматов используется дополнительная спецификация, называемая SMI – Structure of Management Information.

При описании переменных MIB и форматов протокола SNMP спецификация SMI опирается на формальный язык ASN.1, принятый ISO в качестве нотации для описания терминов коммуникационных протоколов (правда, многие коммуникационные протоколы, например IP, PPP или Ethernet, обходятся без этой нотации). Нотация ASN.1 служит для установления однозначного соответствия между терминами, взятыми из стандартов, предназначенных для человеческого использования, и теми данными, которые передаются в коммуникационных протоколах аппаратурой. Достигаемая однозначность очень важна для гетерогенной среды, характерной для корпоративных сетей. Так, вместо того чтобы указать, что некоторая переменная протокола представляет собой целое число, разработчик протокола, использующий нотацию ASN.1, должен точно определить формат и допустимый диапазон переменной. В результате документация на MIB, написанная с помощью нотации ASN.1, может точно и механически транслироваться в форму кодов, характерных для сообщений протоколов.

Имена переменных MIB могут быть записаны как в символьном, так и в числовом форматах. Символьный формат используется для представления переменных в текстовых документах и на экране дисплея, а числовые имена – в сообщениях протокола SNMP. Например, символьному имени SysDescr соответствует числовое имя 1, а более точно 1.3.6.1.2.1.1.1.

Составное числовое имя объекта SNMP MIB соответствует полному имени этого объекта в дереве регистрации объектов стандартизации ISO. Разработчики протокола SNMP не стали использовать традиционный для стандартов Internet способ фиксации численных параметров протокола в специальном RFC, называемом «Assigned Numbers» (там описываются, например, численные значения, которые может принимать поле Protocol пакета IP, и т. п.). Вместо этого они зарегистрировали объекты баз MIB SNMP во всемирном дереве регистрации стандартов ISO, показанном на рис. 3.

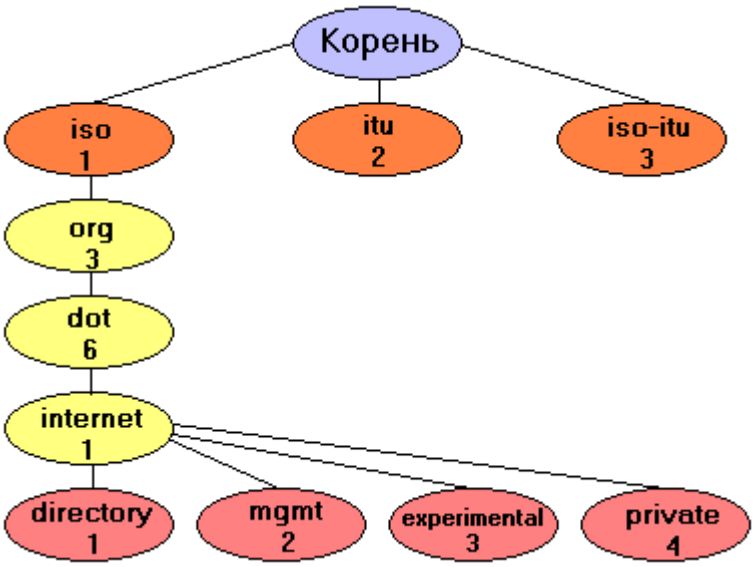


Рис. 3. дереве регистрации стандартов ISO

Как и в любых сложных системах, пространство имен объектов ISO имеет древовидную иерархическую структуру, причем на рис. 3 показана только его верхняя часть. От корня этого дерева отходят три ветви, соответствующие стандартам, контролируемым ISO, ITU и совместно ISO-ITU. В свою очередь, организация ISO создала ветвь для стандартов, создаваемых национальными и международными организациями (ветвь org). Стандарты Internet создавались под эгидой Министерства обороны США (Departament of Defence, DoD), поэтому стандарты MIB попали в поддерево dod-internet, а далее, естественно, в группу стандартов управления сетью – ветвь mgmt. Объекты любых стандартов, создаваемых под эгидой ISO, однозначно идентифицируются составными символьными именами, начинающимися от корня этого дерева. В сообщениях протоколов символьные имена не используются, а применяются однозначно соответствующие им составные числовые имена. Каждая ветвь дерева имен объектов нумеруется в дереве целыми числами слева направо, начиная с единицы, и эти числа и заменяют символьные имена. Поэтому полное символьное имя объекта MIB имеет вид: iso.org.dod.internet.mgmt.mib, а полное числовое имя: 1.3.6.1.2.1.

Группа объектов private (4) зарезервирована за стандартами, создаваемыми частными компаниями, например Cisco, Hewlett-Packard и т. п. Это же дерево регистрации используется для именования классов объектов CMIP и TMN. Именно с веткой private и велась работа в данном проекте.

4. Способ взаимодействия агента и менеджера

В SNMP клиент(агент) взаимодействует с сервером(менеджером) по принципу запрос-ответ. Сам по себе агент способен инициировать только одно действие, называемое ловушкой прерыванием (в некоторой литературе "trap" – ловушка). Помимо этого, все действия агентов сводятся к ответам на запросы, посылаемые менеджерами. Менеджеры же имеют гораздо больший функционал, они в состоянии осуществлять четыре вида запросов:

GetRequest – запрос у агента информации об одной переменной.

GetNextRequest – дает агенту указание выдать данные о следующей (в иерархии) переменной.

GetBulkRequest – запрос за получение массива данных. При получении такового, агент проверяет типы данных в запросе на соответствие данным из своей таблицы и цикле заполняет структуру значениями параметров: for(repeatCount = 1; repeatCount

SetRequest – указание установить определенное значение переменой.

Кроме этого, менеджеры могут обмениваться друг с другом информацией о своей локальной MIB. Такой тип запросов носит название InformRequest.

SNMP – протокол контроля и диагностики, в связи с чем, он рассчитан на ситуации, когда нарушается целостность маршрутов, кроме того, в такой ситуации требуется как можно менее требовательный к аппаратуре транспортный протокол, потому выбор был сделан в сторону UDP.

Но это не значит, что никакой другой протокол не может переносить пакеты SNMP. Таковым может быть IPX протокол (например, в сетях NetWare) , также в виде транспорта могут выступать кадры Ethernet, ячейки ATM. Отличительной особенностью рассматриваемого протокола есть то, что передача данных осуществляется без установки соединения.

5. SNMP-traps.

В SNMP клиент взаимодействует с сервером по принципу запрос-ответ. Сам по себе агент способен инициировать только одно действие, называемое ловушкой прерыванием (в некоторой литературе "trap" – ловушка).

Ловушка включает в себя текущее значение sysUpTime (время), OID (идентификатор объекта, от которого пришло уведомление), определяющий тип trap (ловушки), и необязательные связанные переменные. Адресация получателя для ловушек определяется с помощью переменных trap-конфигурации в базе MIB.

По своей сути trap, как было сказано ранее – это уведомление, реакция на какое-либо событие. Например – превышена температура на устройстве, и оно перегревается.

Заключение.

Была изучена система мониторинга Zabbix в части передачи данных при помощи протокола SNMP. В процессе изучения вопроса модификации имеющейся системы были разработаны MIB (12 шт.), позволяющие обрабатывать сигналы об авариях, которые проходят через первичные мультиплексоры.

Библиографический список

1. SNMP протокол – принципы, безопасность, применение. – http://www.codenet.ru/webmast/snmp/ (дата обращения, 01.12.2020)
2. SNMP-trap (ловушки SNMP) – https://www.10-strike.ru/lanstate/help/snmp-trap.shtml (дата обращения, 01.12.2020)
3. Free Online MIB Database – http://www.oidview.com/mibs/detail.html (дата обращения, 03.12.2020)
4. Для чего предназначен SNMP – https://ip-calculator.ru/blog/ask/dlya-chego-prednaznachen-snmp-rukovodstvo-po-nms-mib-oid-lovushkam-i-agentam/ (дата обращения, 05.12.2020)
5. https://www.zabbix.com/

THE PRINCIPLE OF OPERATION OF THE MONITORING SYSTEM

Sannikov Sergey N.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, serega.sann@mail.ru

Abstract: Monitoring systems are used everywhere in the field of communications. The monitoring function is designed for those who want to get high-level infrastructure monitoring. In most cases, IT infrastructure managers are not interested in low-level details, such as lack of disk space, high CPU usage, etc. They are interested in the availability of the service provided by their IT department. Managers may also be interested in identifying weaknesses in the IT infrastructure, SLA (Service Level Agreement) of various IT services, the structure of the existing IT infrastructure, and other information at a higher level. This study is devoted to studying the capabilities of the Zabbix monitoring system and identifying the prospects for its modification for use in systems based on primary multiplexers. In particular, considerable attention was paid to the

SNMP component of Zabbix and its associated MIB (Management Information Base), which are designed to store and process information about accidents. As a result, 12 MIB was written to handle information about accidents, such as overheating of the boards, lack of signal from the device, high/low input voltage of the device, and more.

Keywords: management system, monitoring system, SNMP agent, SNMP manager.

УДК 004.056.53

ВЫЯВЛЕНИЕ ПРЯМЫХ ПОДКЛЮЧЕНИЙ К TOR В СЕТЕВОМ ТРАФИКЕ ЛОКАЛЬНОЙ СЕТИ

Лихачева София Вячеславовна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, lihacheva_sofia@mail.ru

В данной работе рассматривается задача обнаружения прямых подключений к Tor в сетевом трафике локальной сети. Изучены принципы работы сети Tor, принципы работы TLS, особенности сертификатов TLS, характерные черты сетевых пакетов Tor, исследованы аномалии в сети Tor, рассмотрены и реализованы известные алгоритмы обнаружения прямых подключений к Tor в локальных сетях, выполнено их сравнение, а также сравнение их критериев, была сконфигурирована локальная сеть для тестирования, а также разработана и протестирована программа для обнаружения прямых подключений к Tor в локальной сети. Полученные результаты проанализированы, на их основе разработан алгоритм для обнаружения прямых подключений к Tor в сетевом трафике локальной сети, алгоритм протестирован, его результаты приводятся.

Ключевые слова: Tor, сетевой трафик, анализ трафика, сеть, TLS.

В современном мире анонимность является острым вопросом.

Tor – это бесплатное ПО, предоставляющее возможность установления анонимного сетевого соединения. Благодаря своим возможностям по анонимизации, Tor стал удобным инструментом для преступников, поэтому выявление Tor трафика в сети становится важной частью ее защиты от внешних вторжений и предотвращения утечек информации.

Основная идея работы Tor – перенаправление интернет-трафика клиента через несколько серверов-посредников, или узлов (node, relay).

Данные клиента не раскрываются благодаря луковой маршрутизации – технологии анонимного обмена информацией. Однако из-за ее особенностей каждому клиенту нужно иметь список всех узлов сети.

При работе с Tor используется протокол TLS, поэтому при подключении к сети клиенту отправляется сертификат TLS.

Обычно сертификаты предоставляются удостоверяющими центрами и формируют цепочки, однако они могут быть самоподписанными – когда сертификат является единственным в цепочке и УЦ является сам владелец.

Сертификаты TLS содержат в том числе следующие поля, которые могут идти в произвольном порядке:

- Issuer – доменное имя удостоверяющего центра;
- Validity – период времени, в который сертификат действителен;
- Subject – доменное имя владельца;

- SubjectPublicKeyInfo – алгоритм и открытый ключ владельца;
- SignatureValue – цифровая подпись.

Далее перейдем к рассмотрению реализуемых алгоритмов.

Согласно работе [2] атака «Черный список» реализуется посредством получения списка IP-адресов узлов Tor из открытых источников, далее производится сопоставление этого списка со списком адресов, с которыми устанавливают соединение пользователи.

Одной из основных проблем этой атаки являются ложноположительные срабатывания – на одном IP-адресе могут находиться несколько веб-сайтов, часть из которых может не принадлежать сети Tor. Для решения используется проверка по номеру порта. Он должен входить во множество {9001, 9002, 9003, 9004, 80, 9030, 9031, 9032, 9033, 9100, 443}.

Другой способ обнаружения Tor пакетов, описанный в работе [1], использует анализ особенностей TLS сертификата узла Tor:

- Самоподписанный – обычно TLS включает в себя цепочку сертификатов для аутентификации, однако в Tor цепочка состоит из одного сертификата;
- Специфичный порядок полей – в Tor порядок полей постоянный: подпись, удостоверяющий центр, период действия, владелец сертификата, публичный ключ;
- Удостоверяющий центр – адрес, начинающийся с “www.” и кончающийся на “.com”, между которыми от 8 до 20 букв и цифр.
- Владелец сертификата – адрес, начинающийся с “www.” и кончающийся на “.net”, между которыми от 8 до 20 букв и цифр, при этом не совпадающих с полем “удостоверяющий центр”.

Третий способ, описанный в работе [2], помимо анализа особенностей TLS сертификата использует также параметры соединения:

1. Проверить, входит ли порт соединения во множество {443, 9001, 8443, 22, 80, 8080, 9000, 20000, 20001, 20002}.
2. Проверить удостоверяющий центр – должен начинаться с “www.” и кончаться на “.com”, между которыми от 8 до 20 букв и цифр.
3. Проверить размер сертификата – от 400 до 600 байт.

Далее необходимо выполнить тестирование алгоритмов.

Самый простой способ построения сети для тестирования: сеть, состоящая из роутера и двух узлов, подключенных по беспроводному соединению, при этом один из узлов будет прослушивать трафик.

Далее была реализована программа, которая принимает на вход файл формата pcap, из которого извлекаются данные о каждом соединении (IP-адреса, порты) и сертификате согласно описанным алгоритмам.

Выводится результат: данные о соединении и данные сертификата, а также процентное соотношение количества выполненных условий к общему числу условий для каждого метода.

Далее будут представлены результаты в удобной для изучения форме. Данные в таблицах 1-3 приведены в сокращенном виде в целях отображения общей тенденции.

Таблица 1. Результаты анализа пакетов согласно первому алгоритму

IP получателя	порт получателя	результат, %
91.143.87.51	80	75
100.14.159.254	9001	100
100.14.159.25	9001	100
91.143.87.51	80	75
62.138.7.171	8001	50

Кроме проблем, указанных при описании алгоритма, обнаружилась проблема неполного списка портов.

Таблица 2. Результаты анализа пакетов согласно второму алгоритму

длина цепочки	владелец	удостоверяющий центр	порядок полей	итог, %
1	www.mafmxpgns.net	www.lsnyfwev5634p.com	п_1	100
1	www.sfx4ysjuzsiw3.net	www.qdvdzo7vm43.com	п_2	75
1	www.3eqcby7k7sd.net	www.gqo5nqozpxj.com	п_2	75
1	www.zorygopdh.net	www.5hvwvif3t.com	п_1	100
1	www.k7cxqrxhcqca.net	www.aquoxkryiag.com	п_2	75

Условные обозначения:

п_1 – подпись, УЦ, период действия, владелец, публичный ключ;

п_2 – УЦ, период действия, владелец, публичный ключ, подпись;

Порядок полей сертификата часто не совпадает с заявленным в статье – примерно в 60% исследованных сертификатов встречается порядок п_2.

Таблица 3. Результаты анализа пакетов согласно третьему алгоритму

порт получателя	размер	удостоверяющий центр	итог, %
80	578	www.lsnyfwevvb75634p.com	100
9001	588	www.qdvdzo7v3aemim43.com	100
443	582	www.r3w7ps2pdhkl.com	100
9901	584	www.gvjuqhjo6e2mi.com	66
9010	571	www.5hvwvif3t.com	66

Также присутствует проблема неполного списка портов.

При анализе отдельных условий выяснилось, что ложных срабатываний не показывают условия на: УЦ и владельца, размер цепочки, размер сертификата.

Данные выводы дают возможность предложить свой алгоритм обнаружения пакетов, основанный на следующих условиях:

1. Удостоверяющий центр – адрес, начинающийся с “www.” и кончающийся на “.com”, между которыми от 8 до 20 букв и цифр;
2. Владелец сертификата – адрес, начинающийся с “www.” и кончающийся на “.net”, между которыми от 8 до 20 букв и цифр, при этом не совпадающих с полем “удостоверяющий центр”.
3. Проверить, является ли сертификат самоподписанным.
4. Проверить размер сертификата – от 400 до 600 байт.

Далее приводятся сравнительные результаты работы четырех алгоритмов на всем массиве проанализированных данных.

Таблица 4. Сравнение четырех алгоритмов

результат алгоритма 1, %	результат алгоритма 2, %	результат алгоритма 3, %	результат алгоритма 4, %	Принадлежит ли сертификат сети Tor?
0	25	33	0	нет
75	100	100	100	да
100	75	100	100	да
100	100	100	100	да
75	75	100	100	да
75	100	100	100	да
100	75	100	100	да

результат алгоритма 1, %	результат алгоритма 2, %	результат алгоритма 3, %	результат алгоритма 4, %	Принадлежит ли сертификат сети Tor?
0	0	33	0	нет
0	0	33	0	нет
75	75	100	100	да
100	75	100	100	да
75	75	100	100	да
50	75	66	100	да
50	75	66	100	да
50	100	66	100	да
100	75	100	100	да
75	75	100	100	да
75	75	100	100	да
100	100	100	100	да
75	75	100	100	да
100	75	100	100	да
75	100	100	100	да
75	100	100	100	да
0	0	33	0	нет
0	25	33	0	нет
0	25	33	0	нет
100	75	100	100	да

Таким образом, алгоритм для выявления прямых подключений к Tor в локальной сети разработан, данные проанализированы. Можно сделать вывод, что алгоритм, разработанный в ходе выполнения работы, зачастую является более точным, чем рассмотренные алгоритмы.

Библиографический список

1. *Krawetz N.* Tor 0day Stopping Tor Connections. [Электронный ресурс] URL: https://www.hackerfactor.com/blog/index.php?/archives/888-Tor-0day-Stopping-Tor-Connections.html (дата обращения 14.04.2021)
2. *Lapshichyov V. V.* Algorithm for analyzing and blocking access to the Tor network. In Proceedings of the 12th International Conference of Security of Information and Networks (SIN 2019) – pp. 27-30

DETECTING DIRECT CONNECTIONS TO THE TOR IN THE LOCAL NETWORK

Likhacheva Sofia V.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, lihacheva_sofia@mail.ru

Abstract. The goal of this paper is to explore detection of direct connections to the Tor in the local network. We studied Tor network and its basics, anomalies in the Tor network packages, TLS and its basics, TLS certificates. To test algorithms local network has been configured. We also developed and compared known algorithms for detecting direct connections to the Tor. Based on

results of comparison, an algorithm has been developed to detect the direct connections to the Tor network.

Keywords: Tor, traffic analysis, network, traffic, TLS.

УДК 004.89:004.58

ПОДХОД К РАЗРАБОТКЕ ОНТОЛОГИЧЕСКИ УПРАВЛЯЕМЫХ РЕШЕНИЙ ДЛЯ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ ПОМОЩНИКОВ В ПОДБОРЕ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

Волобоев Сергей Владимирович, Чуприна Светлана Игоревна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, voloboev@psu.ru

В статье представлена концепция подхода для разработки адаптивных онтологически управляемых помощников в процессе выбора подходящего метода машинного обучения. Данное решение включает в себя активную и пассивную интеллектуальную помощь на основе light-weight онтологий, содержащих знания о структуре системы, доступных методах машинного обучения и историю результатов экспериментов с метаданными о параметрах их проведения, а также позволяет автоматически собирать и запускать на выполнение готовое решение по результатам диалога с пользователем. Подобный подход к созданию интеллектуальных помощников позволяет организовать персонифицированную помощь пользователям различной квалификации, учитывая при этом не только ответы текущего пользователя, но и контекст диалога, обогащенный знаниями системы о применимости тех или иных методов машинного обучения для решения определенного класса задач, предыдущий пользовательский опыт и исторические данные из логов системы. Описывается микросервисная архитектура аналитической платформы, реализующей предложенную концепцию.

Ключевые слова: онтология, ontology-driven, интеллектуальный помощник, аналитическая платформа, машинное обучение, микросервисы

Введение

Из-за сложностей в процессе освоения программных продуктов, предназначенных для анализа и обработки больших объемов данных, актуальным представляется разработка интеллектуальных средств автоматизации процесса выбора необходимого метода машинного обучения на принципах self-service [1], что подразумевает минимизацию обращений конечных пользователей к ИТ-специалистам и разработчикам в процессе освоения системы. В качестве таких интеллектуальных средств могут выступать так называемые интеллектуальные помощники (IA, Intelligent Assistant).

Хотя под интеллектуальным ассистентом [2] часто подразумевается голосовой помощник, в данной работе для решения указанных проблем предлагается использовать программных агентов, коммуницирующих с пользователем посредством обмена текстовыми сообщениями на естественном языке (ЕЯ), что в случае активного IA представляется более предпочтительным.

В случае необходимости подбора метода машинного обучения под специфику решаемой задачи и природу анализируемых данных актуальны не только функции IA по замене «пассивной» справочной компоненты активным интеллектуальным помощником, способным первым задавать вопросы, самостоятельно учитывая контекст ЕЯ-диалога. Также

требуются и функции эксперта, способного анализировать не только ответы пользователя относительно особенностей решаемой задачи и его собственных предпочтений, но и использовать в случае необходимости внешние ресурсы, а также накопленный системой опыт [3] для анализа природы пользовательских данных и особенностей решаемой задачи с целью выбора наиболее адекватного метода машинного обучения.

При создании аналитической платформы и IA на её основе нужно учитывать потенциальную расширяемость набора доступных системе функций и подключаемых сторонних ресурсов, что предполагает постоянное обновление используемых интеллектуальными агентами знаний, а это, в свою очередь, влечет необходимость создания управляемого метаданными расширяемого репозитория системных ресурсов (smart-репозитория). Кроме того, нужно решать проблемы, связанные с совместным использованием в рамках одного программного продукта библиотечных функций, написанных на разных языках программирования.

Концепция предлагаемого подхода

В рамках данной статьи под интеллектуальным помощником понимается работающий по типу текстовых чат-ботов управляемый знаниями программный агент, совмещающий в себе функции как активного помощника, способного самостоятельно активизировать диалог с пользователем и задавать ему наводящие вопросы с учетом контекста диалога с целью конкретизации решаемых проблем и формирования предлагаемого решения, так и пассивного обработчика пользовательских поисковых ЕЯ-запросов.

В обзоре [4] в зависимости от того, какие методы используются для выработки полезных советов и предложений, выделяются следующие пять категорий интеллектуальных помощников пользователя при выборе подходящих вариантов решений в задачах аналитики и извлечении знаний из данных (Intelligent Discovery Assistants, IDAs):

1. Экспертные системы (Expert Systems), использующие правила, определенные человеком-экспертом.
2. Системы мета-обучения (Meta-Learning Systems), автоматически извлекающие правила на основе предшествующего анализа данных.
3. Системы, использующие рассуждения на основе прецедентов (Case-Based Reasoning Systems).
4. Системы планирования, использующие AI-планировщиков для создания и ранжирования валидных процессов анализа данных (Planning-based Data Analysis Systems).
5. Инструментальные среды с визуальным графическим интерфейсом для создания пайплайнов анализа данных и инструменты на основе скриптов, позволяющие автоматизировать проектирование и разработку всех или части шагов процесса анализа данных (Workflow Composition Environments, WCE).

Как следует из аналитических обзоров Gartner [5], ведущего мирового консалтингового агентства в сфере IT, большинство популярных современных платформ анализа данных, попавших в “Магический квадрант Data Science и платформ машинного обучения” относятся к категории Workflow Composition Environments. Однако, как отмечается в [4] и подкрепляется нашими исследованиями, интеллектуальные помощники этой категории являются «псевдо-IDA», потому что предоставляемое ими окружение предлагает большой набор инструментов, помогающих облегчить разработку аналитических приложений, но не оказывает необходимой интеллектуальной помощи в самом процессе выбора подходящих инструментов. Некоторые WCE, например SAS [6], имеют встроенные возможности, упрощающие работу пользователя без его вмешательства (определение типа колонок в загруженном датасете, автоматический подбор метрик для сравнения алгоритмов, генерация отчётов), другие – представляют собой не только набор алгоритмов, но и предлагают некоторые рекомендации (например, автоматическое подключение, распространение метаданных, проверка правильности перед выполнением, рекомендации оператора и т. д.). Тем не менее, отсутствуют необходимые пользователям, не относящимся к

разряду Data Scientist, высокоуровневые IA, помогающие в форме активного диалога ориентироваться в предоставленном инструментальном окружении и выбирать подходящие инструменты с учетом природы анализируемых данных и специфики решаемой задачи.

Предлагаемое нами решение основывается на light-weight (легковесных) онтологиях, за счёт которых и работает активная и пассивная интеллектуальная помощь. Также указанные онтологии используются системой для автоматической генерации высокоуровневого интерфейса, запуска методов, подключения новых модулей, хранения метаданных о структуре датасетов из репозитория системы, а также о проведенных в системе экспериментах с целью выработки подходящих рекомендаций на основе анализа накопленного опыта.

В разрабатываемой системе делается акцент на учёте предпочтений разных категорий пользователей. Опытного специалиста не должны раздражать диалоги с вопросами, ответ на которые кажется ему очевидными. Именно поэтому в системе, помимо активного, предусмотрен пассивный IA, который даёт возможность изучать функции системы, исходя из предпочтений пользователя, отраженных в его поисковом запросе.

Работа активного помощника управляется прикладной онтологией, описывающей знания системы о том, какие методы представлены в репозитории системы, для каких задач машинного обучения подходит тот или иной метод, на каких языках написаны и где хранятся программные реализации этих методов. На рис. 1 представлен фрагмент такой онтологии.

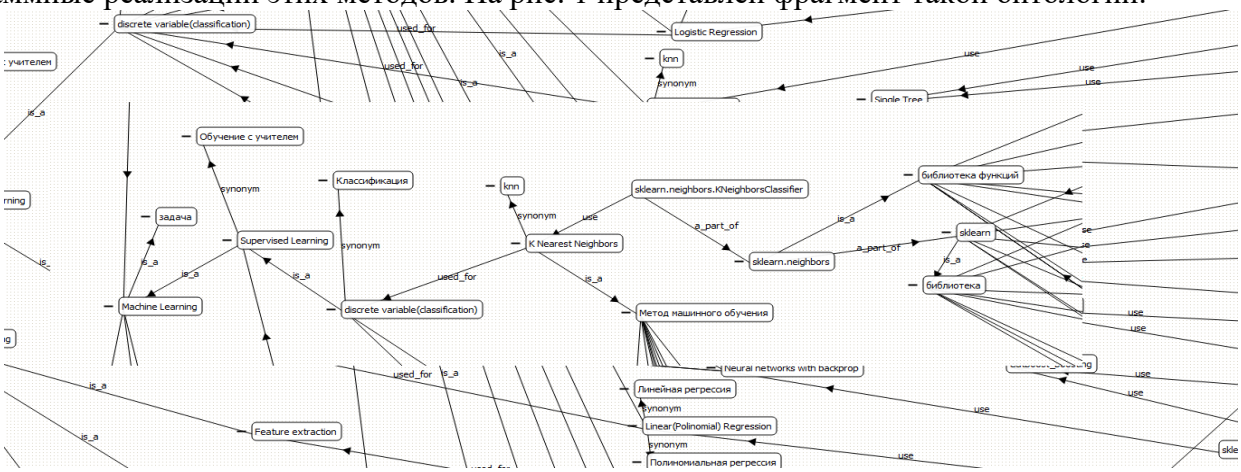


Рис.1 Фрагмент онтологии задач и доступных методов машинного обучения, реализованных на Python

Модель прикладной онтологии поддерживает описание 3-х множеств сущностей: задачи, методы и библиотеки. В правой части рис.1 представлено онтологическое описание алгоритма K-Nearest Neighbors. Связь «is-a» соответствует отношению «класс-подкласс», например:

1. «Classification» is_a «Supervised Learning»,
2. «Supervised Learning» is_a «Machine Learning».

Связь “used_for“ используется для указания взаимосвязи между алгоритмом и решаемой им задачей (в нашем примере – это задача классификации). Связь “use” устанавливает соответствие между методом и реализованным в нём алгоритмом. Для удобства пользователя на уровне интерфейса таксономия задач машинного обучения представлена в виде дерева, где нижележащие понятия связаны с родительскими через связь «is_a». В данной онтологии также поддерживаются синонимы понятий (связь “synonym”).

Так как работа интеллектуального помощника осуществляется под управлением тех же онтологий, что используются для генерации интерфейса, он сразу же, после расширения базы знаний, способен обновлять интерфейс и предлагать пользователю новые решения.

После общения с пользователем IA запускает сборку программных компонентов по принципу “LEGO” и запускает сформированную аналитическую программу на исполнение. Все действия пользователя с системой логируются. Лог содержит записи о выбранных методах и запущенных алгоритмах, метаданные об используемых датасетах и результатах

проведенных экспериментов. Эти метаданные сохраняются в репозитории системы в привязке к соответствующим экспериментам и доступны для дальнейшего анализа с целью сравнения полученных результатов и последующих улучшений рекомендаций IA.

Наш подход предполагает разработку высокоуровневого интерфейса для конечного пользователя, а пополнение и модификация онтологий осуществляется инженером-когнитологом и/или администраторами системы. Для этих целей используется графический редактор Ontolis [7].

Результаты проведённых в системе экспериментов представляются также в формате онтологий и сохраняются в репозитории системы, что позволяет использовать унифицированный механизм для расширения возможностей интеллектуальных помощников как в подборе методов машинного обучения по результатам ЕЯ-диалога с пользователем, так и на основе анализа результатов уже выполненных экспериментов и их сопоставления с текущими потребностями пользователя. Причем активную роль и в том, и в другом случае играет IA.

Чтобы иметь возможность динамически реконфигурировать систему и запускать методы из различных сред, предлагается использовать микросервисную архитектуру [8]. Ключевые модули системы и их интерфейсы описаны в соответствующей прикладной онтологии и представлены в виде микросервисов, общающихся по REST-протоколу. При запуске очередного метода система автоматически генерирует спецификацию для образа Docker и запускает его на выбранных данных. Подобный подход позволяет учитывать специфику предметной области, запуская методы машинного обучения на принципах виртуальной интеграции [9].

Данная архитектура представлена на рис. 2. На ее основе реализован управляемый онтологиями, описывающими методы и средства машинного обучения на языках Python и R, демо-прототип интеллектуального помощника.

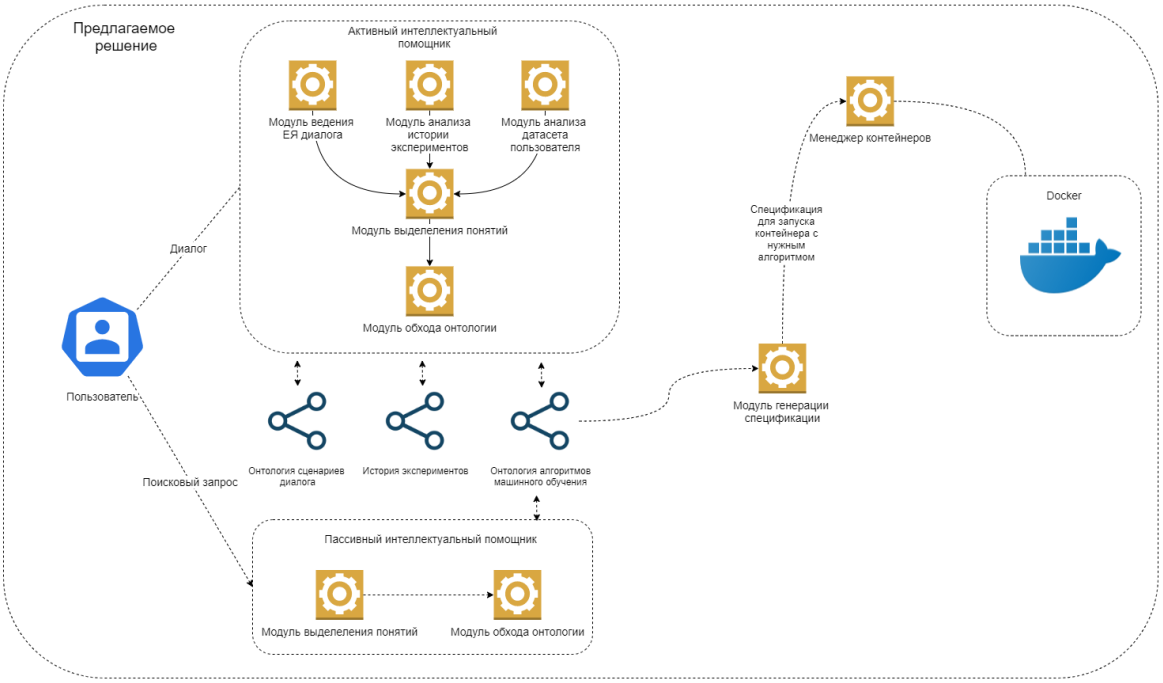


Рис. 2 Микросервисная архитектура аналитической платформы

Дальнейшие исследования планируется посвятить расширению онтологической базы знаний системы и внедрению продвинутых средств визуальной аналитики с использованием инструментов платформы SciVi [10], что позволит совершенствовать средства активной интеллектуальной помощи без необходимости внесения изменений в исходный код ядра аналитической платформы. Планируется также усовершенствовать предложенную архитектуру с целью эффективного масштабирования системы для случая высоконагруженных проектов.

Библиографический список

1. *Meuter ML, Bitner MJ, Ostrom AL, Brown SW.* Choosing among Alternative Service Delivery Modes: An Investigation of Customer Trial of Self-Service Technologies. // Journal of Marketing. 2005. P. 61-83.
2. Обзор персональных цифровых ассистентов 2020: на пути к контекстной адаптации [Электронный ресурс] URL: https://actcognitive.org/storage/uploads/docs/Обзор_персональных_цифровых_ассистентов_2020.pdf (дата обращения: 20.04.2021).
3. *Rafailidis D., Manolopoulos Y.* Can Virtual Assistants Produce Recommendations? // Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics. 2019. P. 1-6.
4. *Serban, F., et al.* A survey of intelligent assistants for data analysis. // ACM Comput. Surv. 45(3). 2013. P. 1–35. https://doi.org/10.1145/2480741.2480748
5. Magic Quadrant for Data Science and Machine Learning Platforms [Электронный ресурс] URL: https://www.gartner.com/doc/reprints?id=1-25D1UHYX&ct=210302&st=sb (дата обращения: 21.04.2021).
6. SAS® Visual Data Mining and Machine Learning [Электронный ресурс] URL: https://www.sas.com/en_us/software/visual-data-mining-machine-learning.html (дата обращения: 21.04.2021).
7. *Чуприна С.И., Зиненко Д.В.* ОНТОЛИС: адаптируемый визуальный редактор онтологий // Вестник пермского университета. 2013. № 3. С. 106-110.
8. Микросервисы, SOA и API: друзья или враги? [Электронный ресурс] URL: https://www.ibm.com/developerworks/ru/library/1601_clark-trs/index.html (дата обращения: 24.04.2021).
9. *Тузовский А. Ф., Ямпольский В. З.* Интеграция информации с использованием технологий semantic web // Проблемы информатики. 2011. №. 2.
10. *Ryabinin K., Chuprina S.* Adaptive scientific visualization system for desktop computers and mobile devices // Procedia Computer Science. 2013. V. 18. P. 722-731.

ONTOLOGY-DRIVEN FRAMEWORK TO CREATE INTELLIGENT ASSISTANTS CHOOSING MACHINE LEARNING ALGORITHMS

Voloboev Sergei V., Chuprina Svetlana I.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, voloboev@psu.ru

Abstract. We propose a framework for developing adaptive ontology-driven intelligent assistants for the process of machine learning algorithms selection. The present solution combines both active and passive intelligent help based on lightweight ontologies using. The ontologies represent knowledge about the system's structure and logs of experiments' history. Ontology-driven approach for intelligent assistants designing provides not only ability to take into account various qualification of the platform's users, but their previous experience and historical data from logs as well. The activeness of intellectual assistant facilitates the users to explore various functions of the proposed platform, and the system provides the best-suited method for their purpose. Microservice architecture of the intelligent analytics platform is presented. The architecture allows the intellectual assistants to adapt to modifications of the algorithms' repository in a unified manner.

Keywords: ontology, ontology-driven, intelligent assistant, analytical platform, machine learning, microservices

УДК 004.434

РАЗРАБОТКА СИСТЕМЫ ПОСТРОЕНИЯ ЛЕКСИКО-СИНТАКСИЧЕСКИХ ШАБЛОНОВ НА ОСНОВЕ АНАЛИЗА КОРПУСОВ ТЕКСТОВ

Боровин Владислав Андреевич

Пермский государственный национальный исследовательский университет,
614990, Россия, г. Пермь, ул. Букирева, 15, batonaq@yandex.ru

Одним из способов извлечения информации из текстов является использование шаблонов, основанных на лексических и синтаксических особенностях текста. Ручная разработка лексико-синтаксических шаблонов зачастую является трудозатратным процессом. Подобная проблема возникла при разработке проекта «Автоматизированная проверка текста, написанного на английском языке русскоязычными авторами (ADWISER)», где для проверки текста используются лексико-синтаксические шаблоны, позволяющие оценить принадлежность текста к научному стилю. Предлагается подход, позволяющий упростить решение поставленной задачи на основе автоматизации построения лексико-синтаксических шаблонов. Представлены результаты сравнительного анализа подходов к автоматизации построения лексико-синтаксических шаблонов. Предлагается метод извлечения лексико-синтаксических шаблонов – маркеров научного стиля, основанный на результатах анализа. Выполнены проектирование, выбор методов и инструментальных средств реализации системы построения лексико-синтаксических шаблонов на основе предложенного подхода к извлечению шаблонов. Предложенный подход реализован, приведен анализ полученных результатов.

Ключевые слова: обработка естественного языка, лексико-синтаксические шаблоны, маркеры научного стиля, синтаксический анализ, автоматизация обработки текста.

В связи с растущим количеством неструктурированной информации в сети Интернет, растет и актуальность задач ее обработки и структурирования. В большинстве своем, такая информация представляется в текстовом виде на естественном языке. Эффективно обрабатывать большие массивы текстов можно лишь с помощью автоматизированных средств. Одним из способов извлечения информации является использование шаблонов, основанных на некоторых особенностях текста. Шаблоны, которые могут быть использованы для распознавания лексических и синтаксических особенностей текста, называются *лексико-синтаксическими* [1].

Порой лингвистам в ходе исследований требуется разрабатывать тысячи, десятки тысяч лексико-синтаксических шаблонов, что является очень трудоёмким процессом. Аналогичная проблема возникла и при реализации проекта «Автоматизированная проверка текста, написанного на английском языке русскоязычными авторами (ADWISER)», где лингвистам необходимо разрабатывать шаблоны, позволяющие оценивать принадлежность текстов к научному стилю. Шаблоны разрабатываются вручную, для построения достаточно точных, подходящих для решения конкретных задач исследования шаблонов требуется проанализировать большое количество научных публикаций. Автоматизация извлечения шаблонов из корпусов таких текстов должна упростить решение поставленной задачи.

Задача стилистического оценивания научных публикаций

Для новой идеи, теории, гипотезы основной формой представления является научная статья. Для статей характерен академический стиль, требования соблюдения которого предъявляются издателями. Проблема оценивания научной английской речи (в частности, при подготовке специалистов, в ходе образовательного процесса) является актуальной в современных условиях, учитывая потребность в специалистах со знанием английского языка.

Практически отсутствуют работы, в которых были бы научно обоснованы критерии и нормы оценивания стилистически нормированной научной письменной речи студентов [2]. Существуют различные критерии научного стиля, в том числе стилевое соответствие, которое позволяют оценить *маркеры стиля* – лексико-синтаксические конструкции, свойственные текстам в данном стиле. Чтобы установить стилевое соответствие публикации установленным требованиям, необходимо разработать соответствующие шаблоны, задача автоматизации построения которых рассматривается ниже.

Алгоритм извлечения шаблонов

Различными исследователями [3-6] предпринимались попытки автоматизировать процесс извлечения лексико-синтаксических шаблонов из корпусов текстов на естественном языке в различных предметных областях. Предложенные ранее методы оказались непригодны для использования в рассматриваемой предметной области. Все существующие подходы позволяют найти лишь некий конкретный набор конструкций для дальнейшей обработки, таким образом, требуется разработать новый подход к решению поставленной задачи, адаптируя существующие методы к специфике проекта.

Сравнительный анализ выявил общие шаги в известных алгоритмах извлечения шаблонов: 1) предобработка обучающего корпуса, 2) поиск ключевых слов в корпусе из некоторого словаря терминов, 3) анализ контекста ключевых слов, 4) извлечение конструкций из контекста, 5) извлечение необходимой информации из конструкций.

Основным отличием рассматриваемого проекта является то, что маркеры научного стиля могут встречаться не только в предложениях с ключевыми словами для конкретной предметной области (научно-техническими терминами, например). Следовательно, необходимо рассматривать все существующие конструкции, строя синтаксические деревья и выделяя наиболее распространённые из них, что определяет особенность предлагаемого подхода. Первым шагом предложенного алгоритма извлечения лексико-синтаксических шаблонов является *автоматическая предобработка корпуса*. Далее предлагается *для каждого предложения в корпусе выполнять синтаксический анализ* и строить дерево синтаксического разбора. Полученные деревья трансформируются для получения наиболее обобщенных конструкций с помощью *методов обобщения*.

Извлеченные таким образом шаблоны рассматриваются как *маркеры научного стиля*, поскольку преобладание маркеров определенного стиля над маркерами других стилей (или их полное отсутствие) является одним из критериев принадлежности текста к данному стилю.

Проектирование системы

В соответствии с шагами алгоритма выделены сценарии использования системы. Она должна позволять: загружать текстовые файлы, составляющие корпус; выполнять их автоматическую предобработку; извлекать и обобщать лексико-синтаксические конструкции из предложений корпуса; генерировать лексико-синтаксические шаблоны, соответствующие полученным конструкциям. На основе функциональных требований к системе были выделены четыре *основных модуля* (рис. 1): модуль ввода/вывода, модуль предобработки, модуль извлечения конструкций и модуль генерации шаблонов.

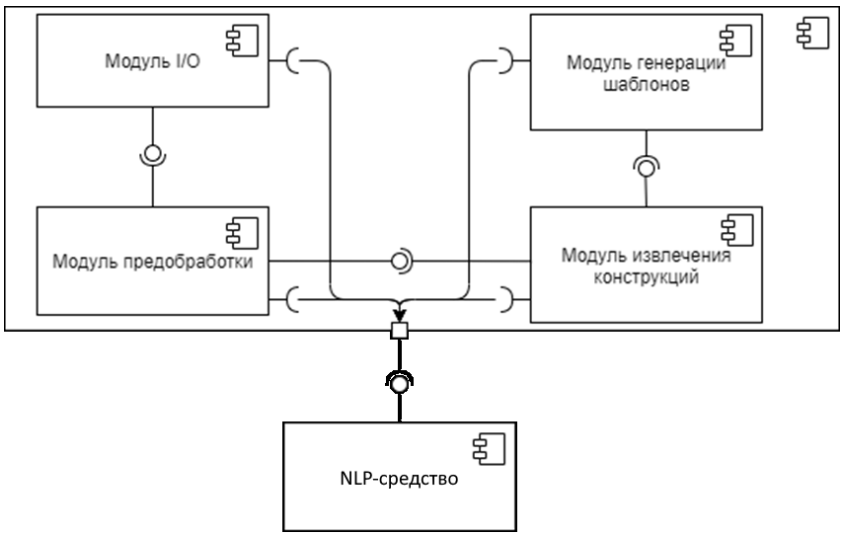


Рис. 1. Диаграмма компонентов системы

Модуль ввода/вывода отвечает за чтение текстовых файлов. Модуль предобработки документов – за подготовку текста к синтаксическому анализу. Модуль извлечения конструкций выделяет лексико-синтаксические конструкции в подготовленном тексте. Модуль генерации шаблонов преобразует выделенные конструкции в шаблоны.

Для реализации системы выбран фреймворк GATE [7], поскольку он интегрирует в себе необходимые модули предобработки текстов. Система реализуется в качестве консольного приложения, связанного с фреймворком GATE с помощью библиотеки GATE Embedded. На вход программа принимает описание конвейера GATE с проектом OpenNLP, включающим шаг синтаксического анализа, и корпус документов, из которых будут извлекаться шаблоны. В свою очередь, извлекаемые конструкции представляются на встроенном в GATE языке описания лексико-синтаксических шаблонов JAPE.

Реализация модуля ввода/вывода

Модуль ввода/вывода (модуль I/O) отвечает за чтение текстовых файлов и создание ресурсов Document и Corpus в GATE. Система принимает на вход описание конвейера GATE и корпус документов. Описание конвейера GATE – сериализованная в XML формате последовательность ресурсов обработки документов. Данную информацию можно сохранить из GATE Developer – пользовательского интерфейса для работы с фреймворком GATE. Модуль I/O также отвечает за вывод результатов работы системы.

После инициализации GATE Embedded выделяется пул потоков, основанный на ExecutorService. В соответствии с заданным количеством потоков создается необходимое количество копий конвейера GATE, каждому из которых назначается собственный объект корпуса. При последовательной обработке документов проверяется наличие свободного потока в пуле. Если таковой имеется, он начинает обработку текущего документа. После завершения обработки поток возвращается в пул.

Реализация модуля предобработки

Модуль предобработки документов отвечает за подготовку текста к синтаксическому анализу. Часть шагов выполняется с помощью встроенных ресурсов OpenNLP: токенизация (лексический анализ), разделение предложений, частеречная разметка, выделение фраз (например, глагольных групп), распознавание именованных сущностей. Кроме того, реализуются дополнительные шаги для нормализации текста.

На данный момент реализована каноническая декомпозиция Unicode (Normalization Form D), а также удаление не-ASCII символов, переносов строк и скобок.

Реализация модуля извлечения конструкций

Модуль извлечения конструкций, используя результаты этапа предобработки, выполняет следующие шаги: синтаксический анализ, обобщение деревьев разбора (лексико-синтаксических конструкций), подсчёт распределения и фильтрация конструкций.

Результат *синтаксического анализа* предложений представляется с помощью аннотаций GATE, которые снова перестраиваются в синтаксическое дерево.

Реализованы следующие шаги обобщения деревьев:

1. В процессе восстановления из дерева удаляются «служебные» вершины, а при наличии нескольких потомков такие вершины разбиваются.
2. Аналогично разбиваются сложные фразы, содержащие несколько подфраз (сложносочиненные, сложноподчиненные).
3. Все подфразы выделяются в отдельные фразы, при этом родительская фраза не удаляется.
4. При наличии нескольких полностью повторяющих друг друга подфраз они объединяются в одну с указанием количества повторений. На данном шаге заканчивается работа параллельных потоков обработки документов. Полученные конструкции помещаются в общий список, выделенный корпус очищается и поток возвращается в пул.

После этапа обобщения деревьев для выделенных конструкций *вычисляется их распределение по всем документам корпуса*. Для сравнения конструкций представляющие их деревья конвертируются в строковое представление. Пример строкового описания конструкции показан на рис. 2.

```
are of the shortest range; (VP (VBP )(PP (IN )(NP (DT )(NN )(NN ))))
```

Рис. 2. Пример строкового описания конструкции

Модуль извлечения конструкций передает на вход модулю генерации набор наиболее распространенных лексико-синтаксических конструкций из предложений, составляющих корпус. При этом деревья синтаксического анализа обобщаются для получения большего количества менее специализированных шаблонов.

Реализация модуля генерации шаблонов

Модуль генерации шаблонов преобразует поступающие лексико-синтаксические конструкции в лексико-синтаксические шаблоны на языке JAPE. На рис. 3 представлен пример шаблона, описанного с помощью JAPE.

```
Rule: Rule
(({Token.string==~"\w+ly"} & {Token.category == RB})
        ({Token.category == VB} & {Token.string==~"\w+ing"}))
:match --> :match.Result = {rule = "Rule"}
```

Рис 3. Шаблон для определения усилительного наречия с глаголом на языке JAPE

Генерируемые шаблоны создают на месте совпадений аннотации с меткой «pttrn». В дальнейшем возможно расширение свойств аннотации, например указание типа конструкции, извлекаемой шаблоном. Пример извлеченных шаблонов и конструкций, которые они описывают, показан на рис. 4.

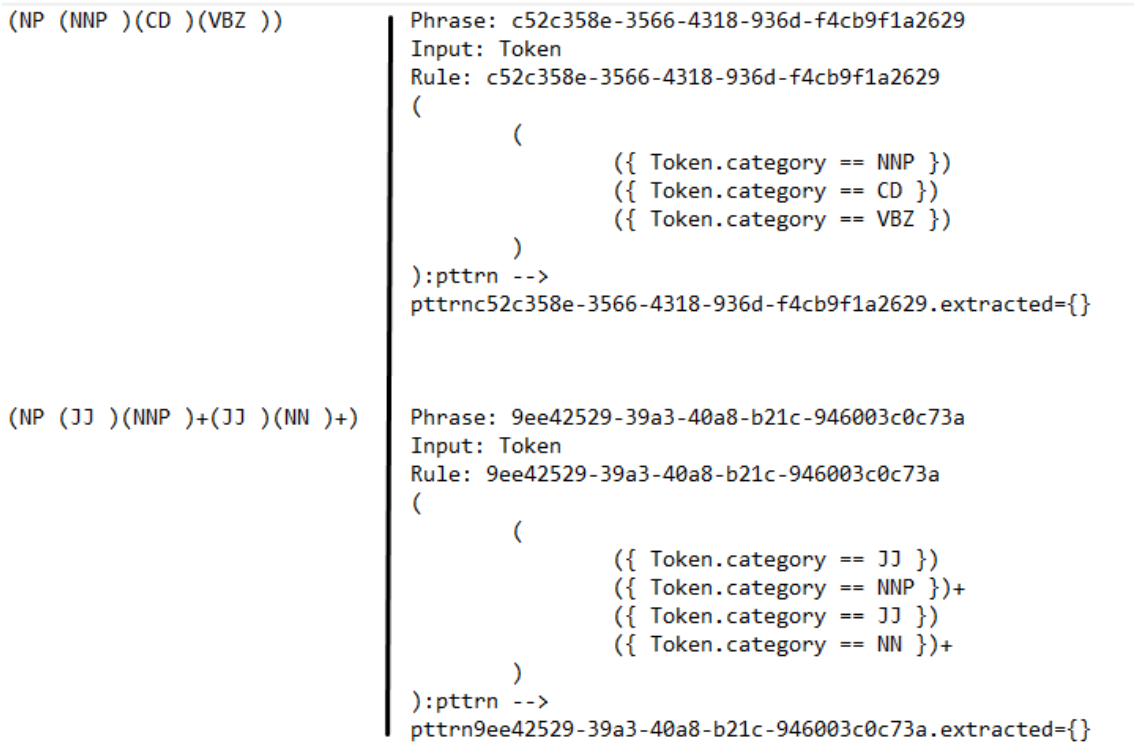


Рис 4. Пример конвертации извлеченной лексико-синтаксической конструкции в описание на языке JAPE

Для представления частей речи JAPE использует Penn Treebank II Constituent Tags [8], поэтому в конвертации данных значений нет необходимости. Фильтр по части речи задается с помощью конструкции «Token.category == часть_речи». Для генерации шаблонов на языке JAPE необходим только листовой уровень синтаксического дерева.

Тестирование системы

Система была протестирована на 49 научных публикациях в открытом доступе по тематике Computer Science. Всего было выделено 118 810 конструкций, 35 662 встретились в единственном экземпляре. Всего было сформировано 6 846 групп, где конструкции встретились более одного раза. Наиболее распространенные конструкции показаны на рис. 5. Диаграммы распределения конструкций показаны на рис. 6, 7.

Время исполнения алгоритма для указанного количества документов без параллельной обработки – 20 минут. После реализации многопоточной обработки время работы сократилось до 6 минут. Наибольшее количество времени занимает синтаксический анализ.

```
(NP (DT )(JJR )(NN ))
(PP (IN )(NP (NP (DT )(NN ))(PP (IN )(NP (JJ )(NN )))))
(VP (VBD )(NP (NNP )(CD )))
(NP (NP (JJ )(NN )+)(PP (IN )(NP (DT )(NN ))))
(VP (VB )(PP (IN )(NP (JJ )(NN )(NNS ))))
(NP (NP (NNP )(POS ))(NN )+)
(VP (VBP )(NP (NN )(NNS )))
(NP (JJ )(NNS )(CC )(NN ))
(VP (VB )(NP (DT )(NN )+(NNS )))
(PP (TO )(NP (NNP )(NNS )))
(PP (IN )(NP (DT )(CD )))
(NP (CC )(NP (NNP )+(CD )))
(NP (NNP )(NN )(NNS ))
(VP (VBZ )(NP (CD )(NNS )))
(NP (PRP$ )(JJ )(NN )(NNS ))
(PP (IN )(NP (NNP )(CC )(NNP )))
(VP (TO )(VP (VB )(NP (JJ )(NN )+)))
(VP (VBZ )(RB ))
(NP (NP (JJ )(NNS ))(WHADVP (WRB )))
(NP (DT )(NNP )(JJ )(NN ))
(VP (VBN )(PP (IN )(NP (JJ )(NN ))))
(ADVP (IN )(JJS ))
(NP (NNP )(NNPS ))
(NP (NNP )(SYM ))
(PP (IN )(NP (JJ )(NN )))
```

Рис 5. Примеры наиболее распространенных конструкций

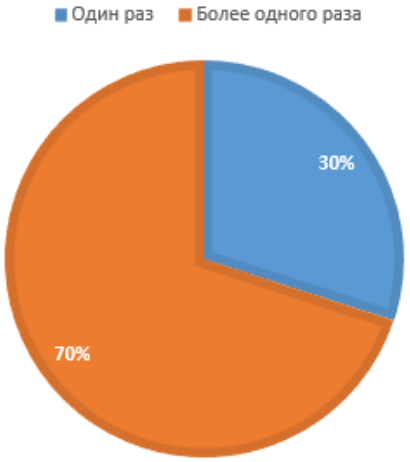


Рис 6. Распределение конструкций по повторяемости

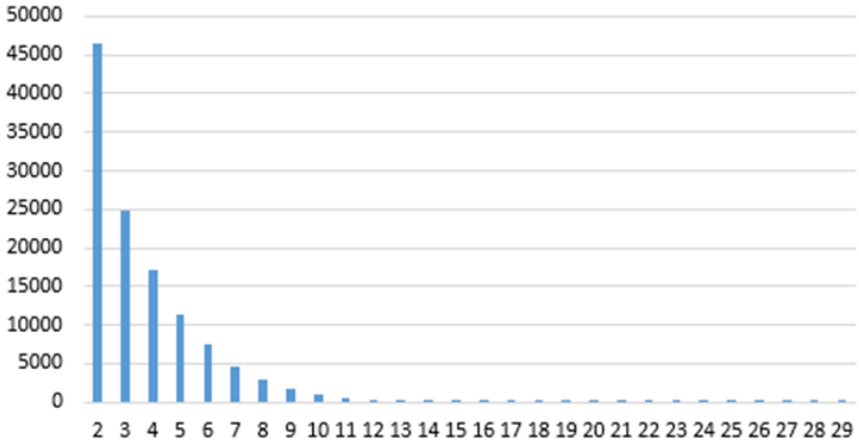


Рис 7. Распределение конструкций по глубине синтаксических деревьев

Некоторые конструкции встретились 300, 400, 500 и более раз. Заметно, что количество конструкций, встретившихся более одного раза, превышает количество конструкций, встретившихся только один раз. В дальнейшем возможно введение этапа объединения конструкций с использованием блоков «ИЛИ» и повторений.

Заключение

Разработанная система позволяет выполнять все выделенные функции и решать поставленную задачу построения лексико-синтаксических шаблонов – маркеров научного стиля – для дальнейшего использования в проекте «ADWISER». Полученное в результате тестирования распределение конструкций показывает хороший уровень обобщения синтаксических деревьев.

Описанный подход можно использовать для извлечения шаблонов из документов в любой предметной области. Использование фреймворка обработки естественного языка GATE позволяет декларативно настраивать конвейер обработки документов корпуса и адаптировать систему к различным условиям и входным данным.

Кроме того, были показаны возможности параллельной обработки корпусов текстов, позволяющей значительно сократить время построения лексико-синтаксических шаблонов.

Библиографический список

1. *Большакова Е.И.*, *Ефремова Н.Э.* Описание языка LSPL // LSPL – Описание проекта [Электронный ресурс]. URL: http://lspl.ru/articles/LSPL_Refguide_13.pdf (дата обращения: 02.04.2021).
2. *Вагнер М.Л.*, *Овезова У.А.* Компоненты, критерии оценки и показатели уровня сформированности умений академической письменной речи студентов-магистрантов. 2019. № 78. С. 292–294.
3. *Apostolova E.*, *Tomuro N.*, *Demner-Fushman D.* Automatic Extraction of lexico-syntactic patterns for detection of negation and speculation scopes // ACL-HLT 2011 – Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol. 2011. Vol. 2. P. 283–287.
4. *Curto S.*, *Mendes A.C.*, *Coheur L.* Question Generation based on Lexico-Syntactic Patterns Learned from the Web // Dialogue & Discourse. 2012. Vol. 3, № 2. P. 147–175.
5. *Gupta S.* Induced lexico-syntactic patterns improve information extraction from online medical forums // J. Am. Med. Informatics Assoc. 2014. Vol. 21, № 5. P. 902–909.
6. *Kirschnick J.*, *Akbik A.*, *Hemsen H.* Freepal: A large collection of deep lexico-syntactic patterns for relation extraction // Proc. 9th Int. Conf. Lang. Resour. Eval. Lr. 2014. Vol. 1. P. 2071–2075.
7. The University of Sheffield. General Architecture For Text Engineering [Электронный ресурс]. URL: https://gate.ac.uk/ (дата обращения: 02.04.2021).
8. Bracketing Guidelines for Treebank II Style Penn Treebank Project [Электронный ресурс]. URL: http://cs.jhu.edu/~jason/465/hw-parse/treebank-manual.pdf (дата обращения: 09.04.2021).

DEVELOPMENT OF LEXICO-SYNTACTIC PATTERN CONSTRUCTION SYSTEM BASED ON CORPORA ANALYSIS

Borovin Vladislav A.

Perm State University, 15, Bukireva St., Perm, 614990, Russia, batonaq@yandex.ru

Abstract. One of the methods of information extraction from natural language texts is to use patterns based on the lexical and syntactic features of the text. The manual development of lexical-syntactic patterns is often a laborious process. A similar problem arose during the development of the project "Automated verification of text written in English by Russian-speaking authors (ADWISER)", where lexical-syntactic patterns are used to check the text, allowing to determine whether the text belongs to the scientific style or not. An approach is proposed to simplify the solution of the problem with automating the lexical-syntactic templates construction. The results of a comparative analysis of approaches and methods to automating the lexical-syntactic templates construction are presented. An approach to extracting lexical-syntactic patterns (scientific style

markers), based on the methods selected as a comparative analysis result, is proposed. The design, selection of methods and tools for the implementation of the system for constructing lexical-syntactic templates based on the proposed approach to template extraction are performed. The implementation of the system is carried out, the analysis of the results obtained is given.

Keywords: natural language processing, lexical-syntactic templates, scientific style markers, parsing, text processing automation.

УДК 004.89

МЕТОДЫ И СРЕДСТВА РАЗРАБОТКИ ЧАТ-БОТОВ НА ПРИНЦИПАХ ОНТОЛОГИЧЕСКОГО ИНЖИНИРИНГА

Бучнев Семен Владимирович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, zxcz@list.ru

В статье представлен подход к разработке чат-ботов на принципах онтологического инжиниринга и методов обработки естественного языка. Концепция рассматривается применительно к виртуальным консультантам, помогающим решать психологические проблемы людей с помощью метода сократического диалога когнитивно-поведенческой терапии. Показана общая схема предлагаемого подхода. Представлены структуры, фрагменты и алгоритмы интерпретации онтологий и применения баз данных для построения персонифицированных чат-ботов психологической помощи. Разработанный демо-прототип мобильного чат-бота демонстрирует работоспособность предложенного подхода.

Ключевые слова: чат-бот психологической помощи, онтологический инжиниринг, методы обработки естественного языка, когнитивно-поведенческая терапия, сократический диалог.

Введение

На сегодняшний день психологические проблемы той или иной степени выраженности встречаются у каждого четвертого-пятого человека в мире [1]. Чаще всего это депрессии, тревожные расстройства и зависимости, протекающие в легких неклинических формах. Люди, которые обнаруживают, что их качество жизни снижено из-за личных психологических проблем, хотят быстро и надежно избавиться от расстройств.

Одним из наиболее эффективных научно-обоснованных подходов к решению ряда клинических и неклинических расстройств является когнитивно-поведенческая терапия (КПТ). КПТ основана на предположении, что в основе психологических проблем человека лежат ошибки мышления, и направлена на изменение нелогичных или нецелесообразных мыслей и убеждений человека. Ключевым инструментом изменения или рационализации мыслей и убеждений является метод сократического диалога, названный в честь древнегреческого философа Сократа, основывающийся на том, что истина не дана в готовом виде и требует поиска. Данный метод предполагает последовательное задание человеку определенных вопросов с целью поиска и исправления ошибок мышления посредством выполнения определенных упражнений [2].

Несмотря на эффективность подхода КПТ для решения целого ряда психологических проблем, возможность его применения на практике существует не всегда. Например, люди могут не иметь денег на дорогостоящие консультации и времени на самостоятельное изучение техник КПТ. Некоторые люди просто не готовы обращаться за помощью к специалистам в области психологии и хотят решать свои проблемы автономно. Для того

чтобы такие люди могли решать свои психологические проблемы, необходимо предоставить им средства, автоматизирующие методы когнитивно-поведенческой терапии.

Средства автоматизации КПТ для клиентов, желающих решать психологические проблемы автономно, могут быть как классическими приложениями, позволяющими вести дневник мыслей и вручную искать логические ошибки, так и виртуальными собеседниками. Виртуальный собеседник или чат-бот – программа, имитирующая диалог с реальным человеком, не требует от пользователя знания концепции когнитивно-поведенческой терапии. Поэтому чат-боты имеют низкий порог вхождения, что значительно увеличивает доступность автоматизированных с их помощью методов КПТ. Чат-боты так же могут интегрировать в себе такие инструменты классических приложений, как ведение дневника мыслей, работа с автоматическими мыслями, просмотр достигнутых результатов.

Распространенные в настоящее время чат-боты психологической помощи предоставляют возможность рационализировать мысли лишь с помощью жестко структурированных, не персонифицированных диалогов, зачастую, на иностранных языках [3]. При этом невозможно проведение полноценного сократического диалога, т.к. не учитывается контекст диалога, а это, в свою очередь, не позволяет в автоматическом режиме более точно интерпретировать семантику фраз клиента и его персональные характеристики для генерации адекватных ответов системы.

Применение методов онтологического инжиниринга в разработке чат-ботов, основанных на когнитивно-поведенческой терапии, позволяет описать базу знаний о психологических проблемах и типичных мыслях конкретного пациента, а также базу знаний о способах ведения диалога для выявления и рационализации негативных мыслей клиентов. В интеграции с методами Nature Language Processing (автоматической обработки текстов на естественном языке) такой подход дает возможность, извлекая из фразы ключевые слова и сопоставляя их с шаблонами из онтологии, выдавать семантически корректный ответ, при необходимости заполняя шаблоны данными клиента.

Концепция предлагаемого подхода

Мы предлагаем использовать управляемый онтологиями подход, предполагающий создание и отладку онтологий [4], которые впоследствии используются для организации диалога с чат-ботом в роли консультанта-психолога. В основе предлагаемого подхода лежит использование легковесных онтологий для описания типичных психологических проблем и их симптомов, упражнений или вариантов диалогов и планов, определяющих упражнения для выявленных проблем. На рис. 1 показана обобщенная схема архитектуры виртуального консультанта, построенного на основе предложенного подхода.

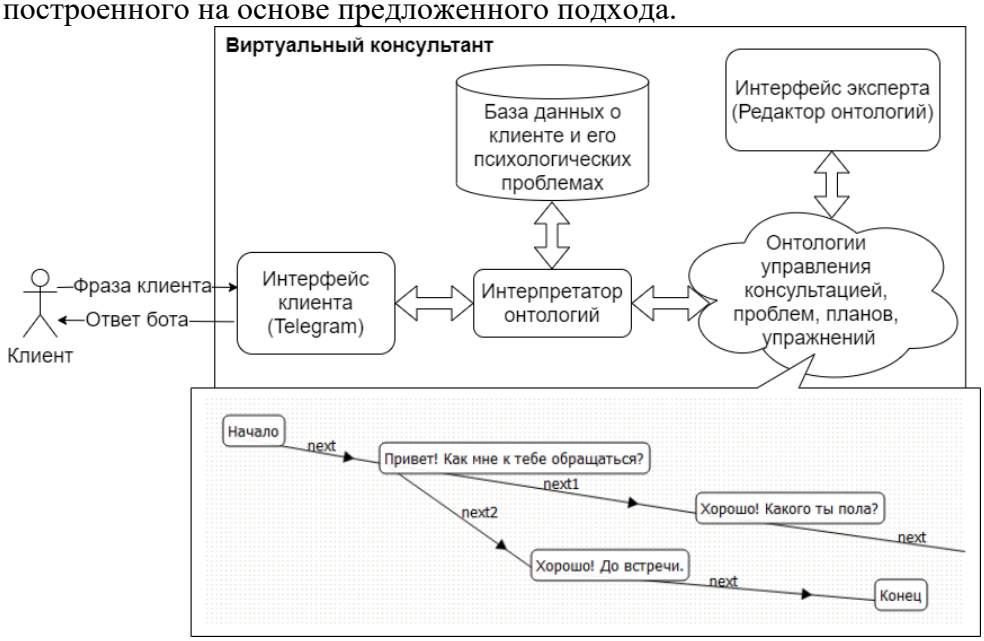


Рис. 1 Обобщенная схема архитектуры чат-бота, основанного на предложенном подходе

Опишем предлагаемый подход более подробно. Клиент с помощью Telegram-интерфейса отправляет чат-боту некоторую фразу на естественном языке (ЕЯ). Фраза клиента попадает в интерпретатор онтологий. Интерпретатор обходит граф онтологии и ищет узлы, соответствующие понятиям из этой фразы. Текст найденных узлов персонифицируется с помощью хранящихся в базе данных о клиенте и выдается клиенту в качестве ответа с помощью Telegram-интерфейса. Построение онтологий осуществляется с помощью интерфейса эксперта, в качестве которого выступает визуальный редактор онтологий ОНТОЛИС, разработанный сотрудниками кафедры математического обеспечения вычислительных систем ПГНИУ [5].

Последовательность шагов (сценарий консультации) задается онтологией управления консультацией, которая легко изменяется/расширяется без необходимости внесения изменений в программный код системы. Эта онтология является онтологией задачи, в которой описываются знания об этапах консультации, порядке их выполнения, соответствующие им упражнения и имена файлов онтологий этих упражнений. Интерпретатор начинает обход графа онтологии с узла «Начало» и движется по этапам с помощью связей next. У каждого этапа есть связанные с ним упражнения, которые доступны по связи «file». На рис. 2 изображен фрагмент онтологии управления консультацией.

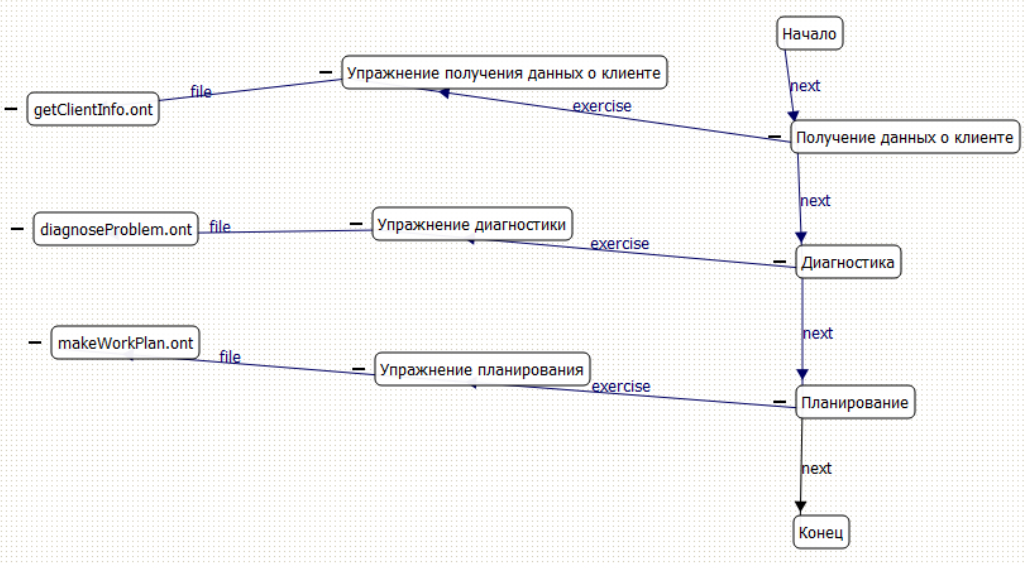


Рис. 2 Фрагмент онтологии управления консультацией

Получение данных о клиенте и упражнения для работы с конкретными проблемами описываются в онтологиях упражнений. Онтология упражнений – это онтология задачи в виде дерева, в котором узлы являются фразами бота, а связи между узлами – возможными типичными вариантами ответа клиента. Варианты ответов пользователя хранятся в соответствующих атрибутах (характеристиках) связи. Интерпретатор начинает обход графа этой онтологии с узла «Начало» и перемещается к первой фразе бота, которую посылает клиенту. Далее в зависимости от настроенного типа взаимодействия бот предлагает выбрать один из нескольких вариантов ответов с помощью нажатия кнопки либо самостоятельно ввести текстовый ответ. В случае, если клиент ввел текстовый ответ, с помощью методов обработки естественного языка определяется, к какому из заданных вариантов ответ клиента ближе с семантической точки зрения. Для этого предлагается использовать классификаторы, построенные на основе языковой модели BERT [6]. BERT представляет собой нейронную сеть, предобученную на большом наборе текстов из Википедии с целью формирования глубинных представлений.

Дообучив модель на сравнительно небольшом объеме данных, можно успешно соотносить ЕЯ-фразу клиента с наиболее близким по семантике вариантом ответа. После того, как интерпретатор определил, какой вариант выбрал клиент, определяется нужная связь и интерпретатор переходит по ней, посылая клиенту следующую фразу. На рис. 1 снизу представлен начальный фрагмент онтологии упражнения для получения данных о клиенте. В

этом случае после фразы бота «Привет! Как мне к тебе обращаться?» у клиента будет возможность назвать свое имя, тогда бот перейдет по связи next1 и продолжит диалог. Если же клиент попрощался, написав, например, «Пока», то бот определит, что клиент желает завершить диалог и, перейдя по связи next2, закончит общение.

Интерпретация ответов клиента и выбор дуги для перехода к следующему шагу диалога выполняется автоматически на основании знаний и фактов, хранящихся в атрибутах узлов и дуг графа онтологии. Вопросы бота, содержащиеся в узлах, могут представлять собой шаблоны, например, «@Client.Name, добрый вечер!», где @Client.Name при выводе ответа заменяется на имя клиента, хранящееся в соответствующем поле базы данных. Онтологии упражнений для решения психологических проблем организованы аналогично и содержат в себе деревья сократических диалогов.

После получения данных о клиенте, для того чтобы определить, какие упражнения нужно выполнять клиенту, проводится диагностика проблем и строится план их решения. В этих целях используются онтологии проблем и планов. Онтология проблем представляет собой прикладную онтологию, содержащую иерархическое описание психологических проблем с присущими им симптомами. Кроме того, она содержит тексты вопросов, которые задаются в процессе диагностики проблем. Интерпретатор начинает обход этой онтологии с самой общей проблемы, узнавая, есть ли подобные симптомы у клиента. Если симптомы обнаружились, то интерпретатор движется глубже в иерархию, уточняя проблему. Если обход дерева завершился или дальнейшие симптомы не подтверждаются, то в базу данных записываются текущие данные о выявленной проблеме клиента. Если ничего не выявлено, то считается, что клиент здоров.

После того, как проблемы выявлены, идет этап планирования работы. В онтологии планов для каждой проблемы указан набор и последовательность упражнений для ее решения. Интерпретатор обходит онтологию планов и на основе хранящихся в базе сведений о проблемах клиента составляет план работы над проблемами. После составления план утверждается клиентом и далее начинается работа по плану. На рис. 3 изображен фрагмент онтологии проблем.

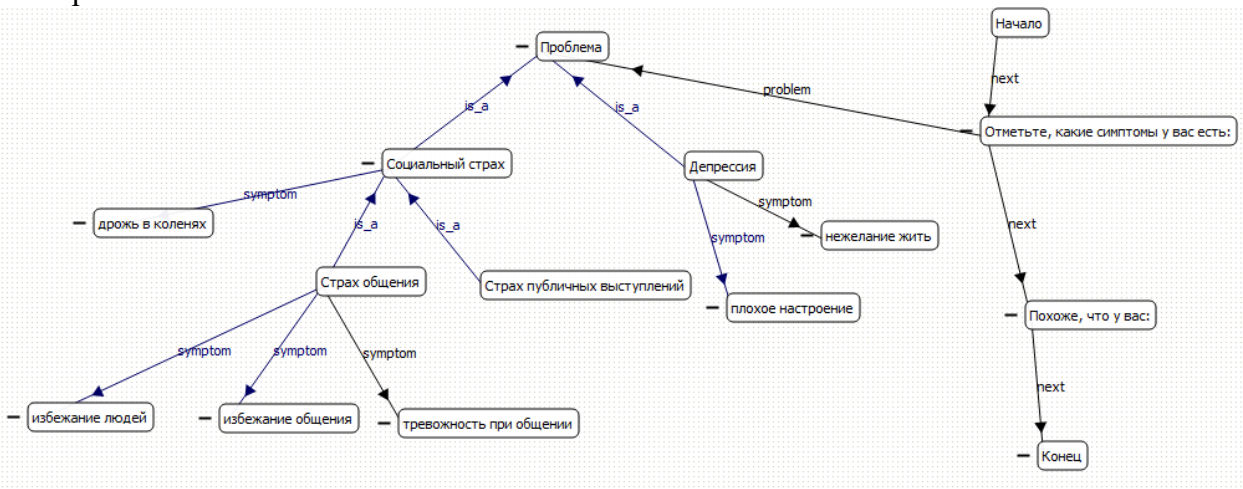


Рис. 3 Фрагмент онтологии проблем

Имея информацию о клиенте, можно достичь хорошего уровня персонификации, поэтому с помощью эксперта-психотерапевта [7] была построена база данных, в которой хранятся данные о клиентах, их проблемах, мыслях, эмоциях, консультациях с ботом и их результатах. Особое внимание требуется уделить безопасности хранения этих данных.

Применение методов онтологического инжиниринга в задаче построения виртуальных консультантов в сфере психотерапии оправдывается возможностью контроля этичности диалогов: онтология наглядна и в ней проще отслеживать нежелательные сообщения. Еще одним преимуществом онтологического подхода является возможность добавления и редактирования знаний о психологических проблемах, планах и упражнениях без изменения исходного кода приложения. Интеграция методов онтологического

инжиниринга, баз данных и обработки естественного языка позволяет создавать персонифицированных виртуальных консультантов, помогающих решать неклинические психологические проблемы клиентов с помощью сократического диалога. В качестве демонстрации работоспособности предложенного подхода на его основе разработан демо-прототип виртуального психологического консультанта.

В ходе дальнейших исследований планируется как расширение спецификации предложенных онтологий и базы данных, так и дальнейшее их заполнение с помощью экспертов. Предлагается подробнее исследовать возможность применения языковых моделей на основе BERT в целях более точного определения семантики фраз клиента, а также автоматизировать построение онтологий на основе Ontology Learning [8].

Библиографический список

1. The World Health Report 2001: Mental Disorders affect one in four people [Электронный ресурс]. Режим доступа: https://www.who.int/news/item/28-09-2001-the-world-health-report-2001-mental-disorders-affect-one-in-four-people
2. Психологическое консультирование и психотерапия: технология сократического диалога [Электронный ресурс]: учеб. пособие / М. В. Бурдин, Е. С. Игнатова; Перм. гос. нац. исслед. ун-т. – Электрон. дан. – Пермь, 2019. – 1,23 Мб; 88 с. – Режим доступа: http://www.psu.ru/files/docs/science/books/uchebnieposobiya/psikhologicheskoe-konsultirovanie-i-psikhoterapiyatehnologiya-sokraticheskogo-dialoga.pdf. – Загл. с экрана.
3. Abd-alrazaq A. A. et al. An overview of the features of chatbots in mental health: A scoping review //International Journal of Medical Informatics. – 2019. – С. 103978.
4. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. – СПб.: Питер, 2000. – 384 с.
5. Чуприна С. И., Зиненко Д. В. ОНТОЛИС: адаптируемый визуальный редактор онтологий //Вестник Пермского университета. Серия: Математика. Механика. Информатика. – 2013. – №. 3 (22).
6. Devlin J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding //arXiv preprint arXiv:1810.04805. – 2018.
7. Бурдин М. В. Индивидуальная кейс-концептуализация в КПТ: варианты и перспективы //ББК 88.4 К57. – С. 16.
8. Somodevilla García M., Vilariño Ayala D., Pineda I. An overview of ontology learning tasks //Computación y Sistemas. – 2018. – Т. 22. – №. 1. – С. 137-146.

METHODS AND TOOLS FOR DEVELOPING CHAT-BOTS BASED ON ONTOLOGICAL ENGINEERING PRINCIPLES

Buchnev Semyon V.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, zxcz@list.ru

The article presents an approach to the development of chatbots based on the principles of ontological engineering and natural language processing methods. The concept is considered in relation to virtual consultants, which help solve psychological problems of people using the method of Socratic dialogue of cognitive-behavioral therapy. The architecture of the proposed approach is demonstrated. Structures, fragments and algorithms for the interpretation of ontologies and the use of databases for the construction of personalized psychological chatbots are presented. Demo prototype of ontology-driven mobile virtual consultant demonstrates the workability of the proposed approach.

Keywords: chatbot, ontological engineering, natural language processing methods, cognitive-behavioral therapy, Socratic dialogue.

УДК 004.434:004.94+004.054

МЕТОД ВЕРИФИКАЦИИ ДИАГРАММ КЛАССОВ UML И ТИПИЧНЫЕ ОШИБКИ СТУДЕНТОВ

Власов Дмитрий Игоревич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, dima.vlasov@icloud.com

Дацун Наталья Николаевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, nndatsun@inbox.ru

Унифицированный язык моделирования (UML) – визуальный язык для проектирования и внедрения программных систем. На этапе проектирования информационных систем создание UML диаграмм позволяет рассмотреть задачу с разных сторон, быстро разобраться во всех взаимосвязях элементов, а также коротко и понятно описать возможности проектируемого объекта. Процесс верификации UML диаграмм может занять достаточное количество времени, а некоторые из ошибок трудно идентифицировать. Во время обучения в университете студенты нередко допускают ошибки, преподавателям приходится проверять несколько сотен диаграмм, поэтому важно проводить процесс верификации быстро и качественно, что сэкономит время на проверку работ и повысит эффективность преподавания и изучения UML диаграмм. В данной работе представлен анализ существующих технологий для верификации диаграмм классов (CD), а также описание разработанного метода проверки CD. Для демонстрации работы выбранного подхода были использованы диаграмм классов, созданные студентами. Показано, что далеко не все идентифицированные ошибки совпадают с ожидаемым результатом. Результат верификации диаграмм зависит от качества построенных моделей. На основе полученных результатов сделаны выводы о самых распространенных проблемах и ошибках, которые возникают у обучающихся при построении диаграмм классов UML.

Ключевые слова: UML, диаграммы классов, верификация диаграмм, лексический анализ, синтаксический анализ, семантический анализ, ошибки студентов.

Анализ существующих решений

Существуют полностью автоматизированные методы верификации диаграмм классов [1, 2]. В подходах, описанных в [3, 4, 5, 6, 7], модели, созданные на UML, транслируются в промежуточные структуры, и это проводится человеком вручную. Такой подход не подходит в случае учебных моделей, так как необходимо быстро проверять диаграммы студенческой группы, а процесс трансляции занимает намного больше времени, чем сама верификация. Кроме того, человек, проверяющий CD, должен обладать определенными знаниями в этой области. Нами был выполнен анализ некоторых из существующих методов и подходов в верификации диаграмм классов UML:

1) Метод, основанный на переводе CD в модель CSP (Constraint Satisfaction Problem) [2]. Перевод в промежуточную модель и сама верификация выполняются в автоматическом режиме. В данном подходе описывается преобразование диаграммы классов в модель CSP. Схема классов CD определяется как CD = ⟨ Cl, As, AC, G, IC⟩ , где Cl – множество классов, As – множество ассоциаций, AC – множество ассоциативных классов, G – множество обобщений и IC набор

ограничений (графических или текстовых), включенных в CD. Каждый элемент транслируется в набор переменных, областей и ограничений в системе CSP. После всех необходимых переводов данная модель подается на вход специальным библиотекам Eclipse (2000 LoC) и Java classes (11500 LoC), расширенным библиотеками Dresden OCL toolkit. Если у созданной модели CSP существует решение, то верифицируемая диаграмма удовлетворяет всем указанным ограничениям, и она построена верно, в противном случае при построении CD были допущены ошибки.

2) Метод верификации с использованием языка Prolog [7]. Метаданные UML элементов, считанные из XMI файла, переводятся в программный код языка Prolog. Этот программный код выполняется, и отображается результат, свидетельствующий о наличии или отсутствии ошибок в верифицируемой UML диаграмме классов.

Воспроизвести и проверить результаты, представленные в статьях [1 – 7], невозможно, так как в них дается только краткое высокоуровневое описание разработанных алгоритмов, а программные средства, их реализующие, не находятся в открытом доступе. Анализ показал, что верификация CD – это сложный процесс, и информации об автоматизированных методах недостаточно, чтобы применить ее на практике.

Описание разработанного метода верификации диаграмм классов

При разработке метода верификации основной задачей являлось определение набора идентифицируемых ошибок, допущенных при построении CD студентами. Этот набор был составлен на основе самых распространенных ошибок при построении CD [8], спецификации UML [9], а также 70 работ студентов ПГНИУ. Эти ошибки были классифицированы нами как лексические, синтаксические и семантические. К лексическим ошибкам отнесены:

1) Использование недопустимых для диаграмм классов элементов;
2) Имена классов, интерфейсов, начинаются с маленькой буквы или содержат пробелы;
3) Имена атрибутов и операций начинаются с заглавной буквы (кроме конструкторов и деструкторов) или содержат пробелы;
4) Имена типов данных не соответствуют именам, используемым в целевом языке программирования и именам других классов модели;
5) Текст различных ограничений не заключен в фигурные скобки.

Группу синтаксических образуют следующие ошибки:

1) У конструкторов и деструкторов указан тип данных результата.
2) У операций (кроме конструкторов и деструкторов) не указан тип данных результата.
3) Класс или интерфейс не содержит ни один параметр и ни одну операцию.
4) Существует класс, интерфейс или перечисление, не имеющее никаких связей.
5) При композиции или агрегации в главном объекте ни у одного из атрибутов в типе данных не содержится имя подчиненного объекта.
6) Ограничения для класса не соответствует атрибутам этого класса.
7) Класс, содержащий члены или операции типа protected, не связан со своими потомками.
8) Между элементом «класс» и «перечисление» связь отличная от типа «зависимость».

Семантическими ошибками являются:

1) Использование при указании кратности ролей ассоциации не целых чисел.
2) Промежуток кратности роли ассоциации начинается с большего числа.
3) Именование обобщения и агрегации.

В основу метода верификации был положен метод CBR [10]. Он представляет метод решения на основе контрольных списков. В нашей работе в качестве контрольного списка используется указанный выше набор идентифицируемых ошибок.

Работа модуля начинается со считывания всех метаданных модели из файла XMI. Все свойства элементов CD могут быть получены из входных данных. Процесс проверки диаграмм аналогичен работе анализатора при компиляции программ на языке программирования и включает в себя три этапа:

1) Лексический анализ.
2) Синтаксический анализ.
3) Семантический анализ.

На каждом из этих этапов идентифицируются ошибки из контрольного списка. На первом этапе считанные данные преобразуются в набор токенов, происходит обнаружение недопустимых элементов для CD, типов элементов, неправильных имен и свойств токенов. На втором этапе проверяется правильность создания конструкций из допустимого набора элементов (связей между элементами CD). На третьем этапе проверки рассматривается семантика построенной диаграммы классов на уровне значений характеристик кратности ассоциаций, правильность значений использованных элементов и их свойств.

Результат тестирования, анализ полученных результатов

Для тестирования модуля проверки CD были выбраны 70 студенческих диаграмм, построенных с помощью инструмента GenMyModel [11]. При тестировании выяснилось, что достоверность результатов зависит от качества построенной и экспортированной модели. 19 диаграмм верифицировать не удалось, так как студенты допускали серьезные ошибки при использовании GenMyModel:

1) Использовали скрытие ненужных элементов вместо удаления.
2) Экспортировали и предоставляли png- и xmi- файлы, не соответствующие друг другу.
3) В специальных полях указывали лишнюю информацию (например, в поле для типа данных переменной указывали еще и имя самой переменной либо наоборот);
4) В одном xmi-файле сохраняли сразу несколько диаграмм (в том числе, не только CD).

Для того чтобы не совершать перечисленные ошибки и получить максимально качественный результат верификации моделей, студентам необходимо ознакомиться с обучающими материалами [12].

Для тестирования модуля была использована 51 диаграмма, созданная студентами. В результате верификации было идентифицировано 1022 ошибки. Распределение ошибок по типам представлено на гистограмме (см. рис. 1). Оно свидетельствует о том, что чаще студенты совершают лексические ошибки. В выбранных для тестирования диаграммах студенты не допустили ни одной семантической ошибки.

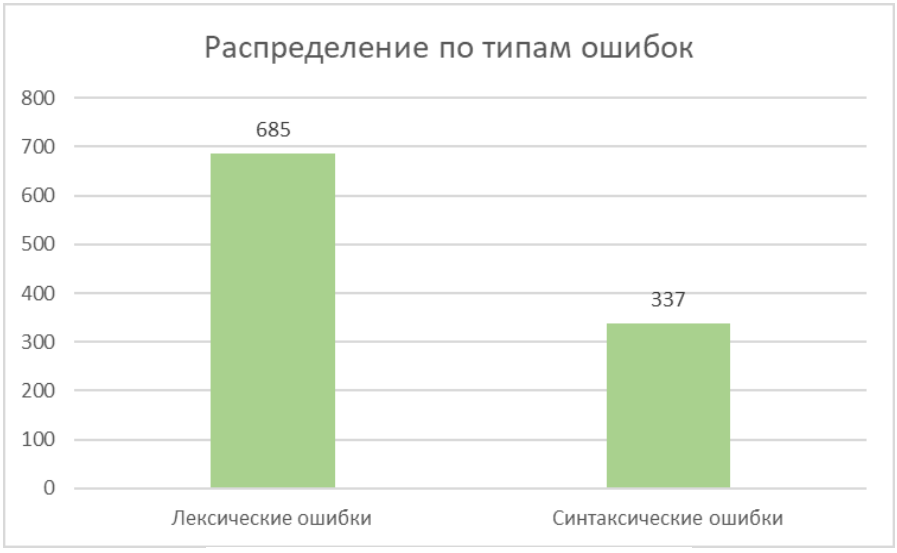


Рис. 1– Распределение по типам ошибок

Top-5 ошибок по частоте встречаемости:

1) Имена типов данных не соответствуют именам, используемым в целевом языке программирования или именам других классов модели (335 ошибок).
2) Ошибки в именах атрибутов и операций (299 ошибок).
3) У операции (кроме конструктора и деструктора) не указан тип данных результата (240 ошибок).

4) При композиции или агрегации в главном объекте должен быть контейнер, тип которого содержит имя подчиненного объекта (49 ошибок).
5) Ошибки в именах классов (42 ошибки).

Большое количество ошибок связано с тем, что студенты допускают одну и ту же ошибку несколько раз в одной и той же диаграмме. Самыми редко встречающимися ошибками оказались следующие:

1) Текст ограничений не заключен в фигурные скобки (9 ошибок).
2) Класс, содержащий члены или операции типа protected, не связан со своими потомками (7 ошибок).
3) У конструкторов и деструкторов указан тип данных результата (4 ошибки).
4) Существует класс, интерфейс или перечисление, не имеющее никаких связей (4 ошибки).

Заключение

В ходе данной работы был проведен анализ существующих методов и подходов верификации диаграмм классов UML. Наличие ограничений у этих методов привело к разработке собственного метода проверки, основанный на CBR.

Проведено тестирование разработанного модуля на 70 диаграммах классов, созданными студентами. Оказалось, что далеко не все идентифицированные ошибки совпадают с ожидаемым результатом. Результат верификации диаграмм зависит от качества построенных моделей с помощью инструмента GenMyModel.

На основе полученных результатов были сделаны выводы об ошибках, которые допускают студенты при использовании инструмента для создания диаграмм, а также было выполнено сравнение количества различных видов ошибок. Выяснилось, что студенты примерно в два раза чаще допускают лексические ошибки, чем синтаксические.

Библиографический список

1. *Berardi D., Calvanese D., Giacomo G.D.* Reasoning on UML class diagrams // Artificial Intelligence. 2005. Vol. 168. pp. 70-118.
2. *Cabot J., Clariso R., Riera D.* On the verification of UML/OCL class diagrams using constraint programming // Journal of Systems and Software. 2014. Vol. 93. Pp. 1-23.
3. *Finite* satisfiability of UML class diagrams by Constraint Programming / M. Cadoli [et al.] // Proc. 2004 International Workshop on Description Logics. 2004. Pp. 1-12.
4. *Jackson D.* Alloy: a lightweight object modelling notation // ACM Transactions on Software Engineering and Methodology. 2002. Vol. 11. Iss. 2. Pp. 256-290.
5. *Queralt A., Teniente E.* Reasoning on UML class diagrams with OCL constraints // Lecture Notes in Computer Science. 200. Vol. 4215. Pp. 497-512.
6. *Gogolla M., Bohling J., Richters M.* Validating UML and OCL models in USE by automatic snapshot generation // Journal on Software and System Modeling. 2005. N 4. Pp. 386-398.
7. *Li Tan, Zongyuan Yang, Jinkui Xie.* UCVSC: A Formal Approach to UML Class Diagram Online Verification Based on Situation Calculus // Proc. 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology. 2009. Pp. 375- 380.
8. *Mistakes* in UML Diagrams: Analysis of Student Projects in a Software Engineering Course / S. Chren [et al.] // Proc. IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training. 2019. Pp. 100–109.
9. *OMG* Unified Modeling Language (OMG UML), Supersrtucture [Электронный ресурс] URL: https://www.omg.org/spec/UML/2.2/Superstructure/PDF (дата обращения: 21.05.2021).
10. *Kösters G., Six H., Winter M.* Validation and Verification of Use Cases and Class Models // Proc. 7th International Workshop on Requirements Engineering: Foundations for Software Quality. 2001. Pp. 1-10.

11. *GenMyModel* [Электронный ресурс] URL: https://app.genmymodel.com/ (дата обращения: 32.05.2021).
12. *GenMyModel tutorial* [Электронный ресурс] URL: https://help.genmymodel.com/en/support/solutions/articles/44000417001-uml-class-diagram-modeling (дата обращения: 21.05.2021).

UML CLASS DIAGRAM VERIFICATION METHOD AND TYPICAL STUDENT MISTAKES

Vlasov Dmitry I.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, dima.vlasov@icloud.com

Datsun Natalia N.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, nndatsun@inbox.ru

Unified Modeling Language (UML) is a visual language for designing and implementing software systems. At the stage of designing information systems, creating UML diagrams allows you to consider the problem from various angles, quickly understand all relationships of the elements, as well as briefly and clearly describe the capabilities of the designed object. UML diagram verification can take a considerable amount of time, and some of the mistakes are difficult to identify. While studying at the university, students often make mistakes, teachers have to check several hundred diagrams, so it is important to carry out the verification process quickly and efficiently, which will save time on checking papers and increase the efficiency of teaching and studying UML diagrams. This paper presents an analysis of existing technologies for verifying class diagrams (CD), as well as a description of the developed verification method. To demonstrate the work of the chosen approach, class diagrams created by students were used It is shown that not all identified errors coincide with the expected result. Verification result depends on the quality of the constructed models. Based on the results, conclusions are drawn about the most common problems and mistakes that students encounter when building UML class diagrams.

Keywords: UML, class diagrams, diagram verification, lexical analysis, syntactic analysis, semantic analysis, student mistakes.

УДК 004.434:004.94+004.052.4

РАЗРАБОТКА АЛГОРИТМА И МОДУЛЯ ВЕРИФИКАЦИИ ДИАГРАММ ДЕЯТЕЛЬНОСТИ, СОЗДАННЫХ СТУДЕНТАМИ

Гашева Татьяна Сергеевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, gasheva99@gmail.com

Дацун Наталья Николаевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, nndatsun@inbox.ru

Формулируется актуальность автоматизации верификации диаграмм деятельности UML, созданных студентами. Приводятся существующие методы верификации этого типа диаграмм UML, обосновывается выбор метода верификации диаграммы деятельностей при помощи графа с семантикой схожей с сетью Петри. Описывается алгоритм выбранного метода и его реализация в модуле верификации диаграммы деятельности, входными данными которого является экспортированный из средства GenMyModel XMI файл, содержащий описание диаграммы деятельности пользователя. Формулируются ограничения, накладываемые на верифицируемые модели. Приводится список ошибок диаграммы деятельности, которые могут быть обнаружены реализованным модулем, а также список ошибок, которые GenMyModel не позволяет совершить пользователю. Приводятся характеристики, результаты анализа и выявленные ограничения разработанного модуля верификации.

Ключевые слова: UML, диаграмма деятельности, верификация, ошибки студентов, XMI.

Введение

UML является широко распространенным стандартом, который предоставляет системным архитекторам, инженерам и разработчикам программного обеспечения инструменты для анализа, проектирования и реализации программных систем, а также для моделирования бизнес-процессов и подобных процессов [1]. Из-за высокой популярности данного стандарта подготовка ИТ-специалистов также предусматривает ими изучение UML [2]. Однако существующие исследования восприятия студентами изучения моделирования на этом языке показывают, что данный процесс воспринимается трудным как для студентов ИТ направления [3], [4] так и для студентов непрограммистов [3].

Для преподавателей процесс ручной проверки диаграмм студентов также может занять много времени. И это подтверждает, необходимость создания инструмента, позволяющего автоматизировать процесс верификации диаграмм UML.

Диаграмма деятельности и ее элементы

Диаграмма деятельности (activity diagram, AD) – диаграмма, показывающая поток управления и данных от одной деятельности к другой, относится к динамическому представлению системы [5]. Основными элементами AD являются начальное состояние, конечное состояние, активность, условный переход, узел слияния, разветвитель, синхронизатор, комментарий, дорожка участника. Также могут использоваться сигналы и таймеры, однако в данной работе будут рассматриваться лишь основные элементы.

Ошибки, допускаемые студентами при создании AD

Для построения AD был выбран инструмент GenMyModel [6], который позволяет создавать AD, экспортировать XMI файл, содержащий описание диаграммы и файл изображения диаграммы. GenMyModel не позволяет пользователю допустить некоторые ошибки:

1) Переход входит в исходный элемент.
2) Начальное состояние имеет входной переход.
3) Конечное состояние имеет выходной переход.
4) Альтернативы записаны не в квадратных скобках.

Список ошибок, которые обнаруживаются модулем верификации AD (далее «модуль»), был составлен на основании существующих каталогов, содержащих распространенные ошибки студентов [7], [8], [9], а также при непосредственном анализе студенческих работ:

1) Имя активности или участника начинается с маленькой буквы.
2) Первое слово в имени активности, возможно, не является именем существительным.
3) В условии отсутствует знак вопроса.
4) Отсутствует подпись в условии.
5) Отсутствует подпись альтернативы.
6) Переход имеет подпись, не являясь условием или альтернативой.
7) В имени участника используется специальный символ.
8) Дорожка участника не содержит элементов.
9) В диаграмме используется недопустимый элемент.
10) В диаграмме больше одного начального состояния.
11) В диаграмме отсутствует начальное состояние.
12) В диаграмме отсутствует конечное состояние.
13) В диаграмме отсутствуют активности.
14) Элемент имеет больше одного выходящего перехода.
15) Условный переход не имеет альтернатив.
16) Условный переход имеет всего одну альтернативу.
17) Элемент (узел слияния или синхронизатор) имеет всего один входной переход.
18) У элемента (кроме начального состояния) отсутствует входящий переход.
19) У элемента (кроме конечного состояния) отсутствует выходящий переход.
20) Элемент не принадлежит никакому участнику.
21) Имя участника или активности не уникальны.
22) Альтернативы ведут в один и тот же элемент.
23) Переход ведет не в активность, разветвитель или в условный переход (разветвитель должен использоваться в паре с синхронизатором, поэтому после него не должно идти конечное состояние; также не имеет смысла сразу же использовать синхронизатор).
24) Альтернатива ведет в условный переход.
25) Возможно, отсутствие парного синхронизатора.
26) Тупик (возникновение ситуации, когда процесс остановлен; возможно, имеется синхронизатор, который нельзя активировать).
27) Недостижимое конечное состояние (возможно, имеется синхронизатор, который невозможно активировать).
28) У одного условного перехода подписи альтернатив не уникальны.
29) Неверная семантика бизнес-процесса.
30) Несоответствие статуса актора и разрешенных ему действий.
31) Противоречия в бизнес-процессе.
32) Подмена бизнес-процесса на «алгоритм работы с ИС».

Эти ошибки делятся на лексические (1-9), синтаксические (10-27) и семантические (28-32). Модуль обнаруживает ошибки 1-28.

По серьезности эти ошибки делятся на предупреждающие (Warning), серьезные (Serious) и фатальные (Fatal). Под фатальными ошибками понимаются те, после обнаружения которых верификация прекращается.

Методы верификации AD

Стандарт XMI позволяет CASE-средствам, поддерживающим этот стандарт, работать с UML-моделью, даже если она была построена в другом инструменте. Этот стандарт может быть использован в качестве эквивалента графическому представлению UML [10]. Также в [10] оговаривается, что, хотя проверка XMI-модели и осуществляется, но полная верификация на соответствие семантическим ограничениям не может быть проведена. Сложности верификации исходной модели возникают именно из-за того, что UML использует визуальное представление модели. Задачу верификации можно упростить, использовав переход от графического представления к эквивалентному ему описанию на формальном языке. В ходе исследования были выявлены следующие методы формального описания диаграммы активности UML:

1) Сети Петри [11].
2) Темпоральная логика [12].
3) Конечные автоматы [13].
4) Использование графа с семантикой схожей с сетью Петри (далее «граф») [14].

В спецификации OMG [1] указано, что AD использует семантику схожую с сетями Петри, то есть семантику, основанную на передвижении токенов. Поэтому в качестве метода верификации при реализации модуля был выбран [14].

Алгоритм верификации диаграмм деятельности

Входными данными модуля является XMI файл; выходными – список ошибок с описанием ошибки и координатами элемента, с которым связана ошибка.

Алгоритм верификации AD состоит из четырех шагов:

1) Парсинг входного XMI файла, содержащего модель.
2) Создание внутреннего представления модели.
3) Предварительная верификация.
4) Верификация с построением графа.

Каждый тип элемента AD представлен классом модуля. На первом шаге алгоритма происходит считывание тегов и атрибутов входного XMI файла, создание объектов соответствующих классов модуля и добавление их в список элементов. На этом же этапе проверяется, принадлежит ли тип элемента набору основных элементов AD. Обязательными полями всех классов являются id, список ссылок на предшествующие элементы и список ссылок на последующие элементы. Некоторые классы содержат дополнительную информацию об элементе, например, условие и список альтернатив у условного перехода.

После создания списка всех элементов на втором шаге алгоритма для каждого объекта из этого списка находятся все предшествующие и последующие элементы. Ссылки на найденные элементы добавляются в список ссылок на предшествующие элементы и список ссылок на последующие элементы.

На шаге предварительной верификации выполняется проверка тех ошибок, которые могут быть обнаружены без графа: ошибки 1-24 и 28. Это ошибки, которые не связаны с парным использованием синхронизатора и разветвителя. Проверка на этом шаге состоит в проходе по списку элементов и проверке подписей элементов и подсчете количества входных и выходных переходов у каждого из элементов.

На четвертом шаге алгоритма ошибки парного использования синхронизатора и разветвителя проверяются с помощью графа.

Для этого шага введем ограничения, накладываемые на входную модель.

Используется набор основных элементов AD, без сигналов и таймеров. Отметим, что элементы комментария и дорожки участника не рассматриваются на этом шаге верификации, поскольку они не влияют на проверку парного использования синхронизатора-разветвителя.

Ограничения на количество переходов между элементами:

1) Начальное состояние не имеет входного перехода.
2) Конечное состояние не имеет выходного перехода.
3) Активность, узел слияния, синхронизатор и конечное состояние могут иметь несколько входных переходов.
4) Активность, узел слияния, синхронизатор, начальное состояние не имеют выходных переходов.
5) Разветвитель и условный переход имеет ровно один входной переход и могут иметь несколько выходных переходов.

При указанных ограничениях невозможна ситуация несвязанных элементов, поскольку все элементы соединены переходами от начального до конечного состояния.

Четвертый шаг алгоритма использует выбранный нами метод верификации AD, использующий граф с семантикой схожей с сетью Петри. Верификация происходит посредством моделирования движения токена вдоль дуг графа и проверки достижимости узлов. Токен представляет собой метку, обозначающую текущий достигнутый элемент. Проверка считается оконченной, если были рассмотрены все возможные пути движения токена от начального состояния до конечного или обнаружена ошибка.

Правила для графа следующие:

1) В начале только начальное состояние имеет токен.
2) Активность потребляет один токен и отдает один токен следующему элементу.
3) Конечное состояние потребляет один токен.
4) Условный переход потребляет один токен и отдает один токен одному следующему элементу.
5) Узел слияния потребляет один токен и отдает один токен следующему элементу.
6) Синхронизатор потребляет по одному токену из каждого предыдущего элемента и отдает один токен следующему элементу.
7) Разветвитель потребляет один токен и отдает по одному токену каждому последующему элементу.

Состояние графа на каждом шаге может быть представлен в виде маски, то есть закодирован последовательностью нулей и единиц, где нуль означает, что элемент неактивен (не имеет токена), а единица означает, что он активен. Создается стек текущих масок, стек промежуточных масок и набор использованных масок.

Проверка осуществляется итерационно. На каждой итерации из стека текущих масок берется верхняя маска и обрабатывается. Результатом обработки являются новые маски, которые помещаются в стек промежуточных масок. После получения всех промежуточных масок, они помещаются в стек текущих масок. Однако перед добавлением в стек текущих масок происходит проверка промежуточной маски на ее использование ранее при помощи набора использованных масок и, если маска ранее не использовалась, то только тогда она добавляются в стек текущих масок.

На каждой итерации все существующие токены перемещаются в один из следующих узлов в соответствии с определенными выше правилами для графа. В случае условного перехода токен может активировать произвольный последующий элемент. Таким образом, он генерирует несколько возможных следующих состояний (несколько масок), которые помещаются в стек текущих масок.

Однако проблема непарного использования синхронизатора и разветвителя сохраняется. Например, при возникновении ситуации, когда нескольким разветвителям соответствует только один синхронизатор, в описанном выше алгоритме переход может быть ошибочно активирован. Для обнаружения такого типа ошибки было предложено использование токенов различного цвета с сохранением цветов в стеках токенов. Схожий метод используется в цветных сетях Петри [15]. «Цвет» – это некие дополнительные данные, которые хранятся в стеке цветов токена. Разветвители генерируют уникальный цвет каждый раз, когда через них проходит токен. При проходе через разветвитель выходящие токены «окрашиваются» в одинаковый цвет, то есть в стек цветов для каждого токена помещается

цвет пройденного разветвителя. При попытке активации синхронизатора проверяются цвета на вершине стеков и если все цвета совпадают и все предшествующие элементы активны, то синхронизатор активируется, а из стеков извлекается по одному цвету. Иначе возникает ошибка, и проверка завершается.

Анализ результатов разработки модуля верификации AD

Размер модуля верификации диаграмм деятельности – 83 КБ. Разработанный модуль позволяет обнаруживать 28 видов ошибок. Тестирование было выполнено на 54 диаграммах деятельности. Степень обнаружения ошибок составляет около 93%. Степень обнаружения лексических ошибок – 92%, синтаксических – 100%, семантических – 8%. Процент диаграмм с фатальными ошибками составляет 16%.

Время работы модуля для диаграмм деятельности представлено в таблице.

Таблица. Время работы модуля для диаграмм деятельности

Без использования синхронизатора/разветвителя		С использованием синхронизатора/разветвителя	
Количество элементов	Время, мс	Количество элементов	Время, мс
10	4	13	15
25	13	23	33
53	41	49	128
87	125	71	376

Были выявлены следующие ограничения модуля:

1) Модуль осуществляет первичную проверку AD, однако не может обеспечить должную проверку семантики диаграммы, которая остается за человеком.
2) Результаты работы модуля зависят от качества входных данных, экспортированных в модели из GenMyModel.
3) Большое количество ошибок в модуле верификации AD являются фатальными и требуют немедленного исправления. При обнаружении такого типа ошибок модуль прекращает дальнейшую проверку, поэтому не все ошибки могут быть обнаружены в текущей сессии проверки диаграмм.

Библиографический список

1. *Unified* Modeling Language 2.5.1. (2017). Object Management Group [Электронный ресурс] URL: https://www.omg.org/spec/UML/About-UML/ (дата обращения: 04.04.2021).
2. *Guide* to the software engineering body of knowledge / P. Bourque [et al.] // IEEE Computer Society Press. 1999. V. 16. pp. 35-44.
3. *Insights* in Students' Problems during UML Modeling / R. Reuter [et al.] // In IEEE Global Eng. Educ. Conf. (EDUCON). Porto. 2020. pp. 592-600.
4. *Matyokurehwa K.,* Makoni K. T. Students' Perceptions in Software Modelling Using UML in Undergraduate Software Engineering Projects // International Journal of Information and Communication Technology Education. 2019. Vol. 15. N 4. Pp. 12-24
5. *Буч Г., Рамбо Д.,* Джекобсон А. Язык UML. Руководство пользователя. – М.: ДМК Пресс, 2001. 248 с.
6. *GenMyModel* [Электронный ресурс] URL: https://app.genmymodel.com/ (дата обращения: 19.05.2021).
7. *Mistakes* in UML Diagrams: Analysis of Student Projects in a Software Engineering Course / S. Chren [et al.] // Proc. IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training. 2019. Pp. 100–109.
8. *On* the use of UML documentation in software maintenance: Results from a survey in industry / A. M. Fernández-Sáez [et al.] // Proc. 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems. 2015. Pp. 292-301.

9. *Chytalová K.* Katalog chyb v UML diagramech [Электронный ресурс] URL: https://is.muni.cz/th/dqtso/Katalog_castych_chyb_v_UML_diagramech.pdf (дата обращения: 04.04.2021).
10. *XML* Metadata Interchange (XMI) Specification, 2015. 122 p. [Электронный ресурс] URL: https://www.omg.org/spec/XMI/2.5.1 (дата обращения: 01.12.2020).
11. *Baresi L., Pezzè M.* On formalizing UML with High-Level Petri Nets // LNCS. 2001. Vol. 2001. Pp. 276-304.
12. *Araujo J., Moreira A.* Integrating UML Activity Diagrams with Temporal Logic Expressions [Электронный ресурс] URL: http://ceur-ws.org/Vol-363/paper8.pdf (дата обращения: 04.04.2021).
13. *Raschke A*. Translation of UML 2 Activity Diagrams into Finite State Machines for Model Checking // Proc. 35th Euromicro Conference on Software Engineering and Advanced Applications. 2009. pp. 149-154.
14. *Rafe V., Rahmani A. T.* Formal Analysis of Workflows Using UML 2.0 Activities and Graph Transformation Systems // Lecture Notes in Computer Science. 2008. Vol. 5160. Pp. 305–318.
15. *Jensen K*. A brief introduction to coloured Petri Nets // Lecture Notes in Computer Science. 1997. Vol. 1217. Pp. 203-208.

DEVELOPMENT OF ALGORITHM AND MODULE FOR VERIFICATION OF STUDENT ACTIVITY DIAGRAMS

Gasheva Tatiana S.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, gasheva99@gmail.com

Datsun Natalia N.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, nndatsun@inbox.ru

Abstract. We formulate the urgency of automating the verification of UML activity diagrams created by students. We present the existing methods of verification of this type of UML diagrams and explain the choice of the method for verifying the activity diagram using a graph with semantics similar to the Petri net. The algorithm of the chosen method and its implementation in the activity diagram verification module, the input data of which is an XMI file exported from the GenMyModel tool containing a description of the user's activity diagram, are described. The restrictions imposed on the verified models are formulated. We present a list of activity diagram mistakes that can be detected by the implemented module, as well as a list of errors that GenMyModel does not allow the user to make. The characteristics, analysis results and identified restrictions of the developed verification module are presented.

Keywords: UML, activity diagram, verification, mistakes students, XMI.

УДК 519.71, 004.42, 519.87

СИСТЕМА ПРЕДЛОЖЕНИЯ РАБОТНИКОВ НА ДОЛЖНОСТЬ ДЛЯ НАУЧНЫХ ПРОЕКТОВ

Никулин Максим Валерьевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, nikulin.maxim0009@yandex.ru

Чупин Антон Викторович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15

Институт механики сплошных сред ПФИЦ, 614013, Россия,

г. Пермь, ул. Королёва, 1, chupin@icmm.ru

В современном научном мире проведение исследований или каких-либо других работ является неотъемлемой частью научной деятельности. Основной задачей данной работы является создание для таких работ и исследований систему предложения работника на должность. Данные работы могут организовываться путем использования систем управления проектами, но научные проекты имеют свои особенности. В данной статье рассматриваются научные проекты и их ключевые особенности. Для научных проектов был предложен концепт системы предложения на должность, учитывая их особенности. В ходе изучения проектов было определено, что все ситуации для назначения работников можно разделить на три группы: работник, подходящий для задачи есть и в данный момент свободен, работник, подходящий для задачи есть, но сейчас занят, подходящего работника для задачи нет. Были использованы задача о назначениях, теория массового обслуживания и поиск семантически близких слов соответственно для каждого из возможных вариантов. Подразумевается, что способности людей заранее известны.

Ключевые слова: Научный проект, менеджмент, предложения работника на должность, прикладное программное обеспечение.

Введение

Наука – непрерывно развивающаяся система знаний объективных законов природы, общества и мышления, получаемых и превращаемых в непосредственную производительную силу общества в результате социально-экономической деятельности.

Основная цель науки – познание объективного мира (теоретическое отражение действительности) и воздействие на окружающую среду с целью получения полезных обществу результатов. Наука поддерживается и развивается в результате исследовательской деятельности общества. [1]

В последнее время все чаще на смену термину «управление научными исследованиями» приходит термин «управление научными проектами». Обусловлено это, наверное, следующим. В характерных для настоящего времени условиях динамично изменяющихся требований к результатам научной деятельности возникает необходимость четко выделять результаты этой деятельности, достигаемые к определенному моменту времени, анализировать: какими силами и с какими затратами эти результаты были достигнуты. [2]

Далее стоит определить, что такое проект. Проект – ограниченное во времени целенаправленное изменение отдельной системы с установленными требованиями к качеству результатов, возможными рамками расхода средств и ресурсов и специфической организацией.

После определения «проекта» следующим логичным шагом стоит определить «научный проект». Под научным проектом будем подразумевать ограниченный во времени целенаправленный процесс выработки, теоретической систематизации и применения нового научного знания с установленными требованиями к качеству результатов, расходу ресурсов и специфической организацией.

Прослеживается определенное сходство научных проектов с обычными проектами. Таким образом, научный проект – частный случай общего проекта.

Основными особенностями научных проектов являются: некоммерческая направленность, неопределённость результатов, продолжительные сроки реализации, трудность оценки как планируемых, так и фактических результатов реализации проектов, необходимость комплексного охвата предметных областей и организации информационного обмена, отсутствие аналогий в ретроспективе, узкая специализация участников. [5]

Концепт системы предложения работника на должность

Предлагается для задач, приходящих в систему, выделять компетенции, которыми должны обладать работники для успешного выполнения работ. Встает вопрос, обладает ли конкретный человек данными компетенциями и насколько хорошо. Для решения этой проблемы было решено использовать нечеткие множества, выставляя каждому человеку коэффициент доверия или обладания данной компетенцией. Из необходимых на текущем этапе компетенций формировать вектора с коэффициентами обладания. *Ставится задача оптимизации*, состоящая в выборе работника с наибольшим значением на каждую компетенцию для задачи.

Система имеет список работников с их компетенциями и количеством задач, над которыми конкретный работник работает.

Входными данными для такой системы будут являться задачи с необходимыми компетенциями, которые выполняются одновременно. Для системы необходимы данные о занятости работников и их компетенции. Результатом работы будет оптимальное разбиение работников на задачи.

Система делает запрос в базу данных работников для получения информации: компетенций, количество задач над которыми работник работает, коэффициенты доверия для каждой компетенции. Исходя из этой информации, будет происходить деление.

Так же из-за того, что проект проекты характеризуются своей изменяемостью, система предложения должна адаптироваться к ситуациям и меняться в зависимости от ситуации.

Реализация

Считаем, что есть некая система управления проектами (СУП), для которой требуется добавить функциональность выбора работников на конкретный проект.

Для работы системы на первом этапе необходимо выделить, на чем системы будет базироваться для предложения того или иного человека на должность. Был рассмотрен концепт системы, в которой пользователь сам отмечает, какими способностями он обладает. Система опирается на компетенции, которые каждый работник выставил для себя или полученные другим путем. Конечно, работник мог умышленно или случайно выставить такие навыки, какими он не обладает. В данном случае необходимо ввести величину доверия. Величина доверия показывает точность обладания конкретной компетенцией конкретного работника.

Для того, что бы система была адаптивной к конкретным ситуациям, было принято решение создать конфигурационный файл, в котором хранятся некоторые параметры системы. Данное решение позволит подстраивать систему под необходимые задачи.

Задачи в системе представляются как набор компетенций, для каждой из которых необходим человек обладающий ею. Выбор реализации системы был не случайным, так как при таком подходе можно предусмотреть несколько способов выбора, как должны распределяться работники. В конкретной системе для распределения работников существуют правила: каждой компетенции должен соответствовать человек, имеющий ее, один человек может «закрыть» несколько компетенций.

Для распределения стоит определить, какие существуют ситуации, связанные с работниками. Было замечено, что существует три варианта для покрытия всех компетенций:

- Для текущей задачи есть все свободные работники, обладающие необходимы компетенциями
- Для текущей задачи есть работники, обладающие необходимыми компетенциями, но они работают над другими задачами
- Для текущей задачи нет работников, обладающих необходимыми компетенциями

Данное разбиение позволит охватить огромное количество все возможных вариантов ситуаций, возникающих в системе.

В начале происходит определение ситуации, а после в зависимости от ситуации используется свой алгоритм. Таким образом, для первого и второго случая было решено использовать теорию назначений, а для третьего – поиск похожих по значению других компетенций, которыми обладают работники.

Для каждого варианта необходимо будет предложить свой подход, по которому будет совершаться разбиение всех участников согласно их компетенциям.

Система имеет микросервисную архитектуру. Такой подход позволит включать систему предложения работника на должность в уже существующие СУП. Кроме того, модули легко заменяемы, а как следствие систему можно подстраивать под свои нужды.

Задача о назначениях

Первый вариант работы программы, описанного в реализации, очень похож на задачу о назначениях, решаемую с помощью венгерского алгоритма.

Изначально назначим работникам задачи согласно коэффициенту доверия, насколько это возможно согласно венгерскому алгоритму. Если по всем задачам удалось, то заканчиваем разделение (такое случается достаточно редко). В противном случае, требуется применить модификацию алгоритма, идея чего заключается в следующем. Когда два человека с одинаково высоким показателем доверия владеют данной компетенцией, одному из работников нужно будет подыскать иную свободную задачу, так же наиболее соответствующую его способностям. Если при текущих условиях не найдется свободной задачи, то нужно будет попробовать подыскать среди задач, предварительно немного занизив наши требования. Таким образом, получим еще 1 вариант разбиения. Как бы искусственно занизить доверие. «Поменяем» по цепочке паросочетания, после чего будет +1. И продолжим назначать таким вот оптимальным образом, пока всем не подыщем работу.

В промежуточном итоге известен всего 1 параметр – коэффициент доверия, который присвоен парам «работник-компетенция» (вес ребер в графе назначений), но можно ввести искусственно 2 дополнительных параметра (веса вершин):

- «производительность» разработчика в условных человеко-часах, которая показывает, насколько эффективно задействован работник
- «сложность» задач. Измеряется тоже в человека-часах. Будет показывать, сколько человеко-часов будет необходимо для решения части задачи с данной компетенцией. Тем самым, если решение части задачи займет у человека с производительностью 8 человеко-часов, но сложность задачи будет, например, 3 человеко-часа, то данного работника можно назначить на еще 1 задачу, где «сложность» будет не больше 5.

Доступный нам коэффициент доверия разделим на сумму «производительности» работника и «сложности» задачи. Таким образом, коэффициент доверия будет показывать не только обладание компетенцией, но и способность к ней.

Изначально мы ничего не знаем о «сложности» задачи и будем рассчитывать у рабочих максимально возможную «производительность». Далее мы будем уменьшать «производительность», но наращивать «сложность» (с учётом того, чтобы их сумма не превышала исходное значение коэффициента), разделяя таким образом работников по задачам.

Заметим, что «венгерский алгоритм» работает, только если компетенций поровну с рабочими. Поэтому отбираем свободных рабочих, которые имеют все нужные компетенции и их количество равно количеству компетенций. Это можно использовать как отправную точку для метода. Стоит заметить, что таким образом будут выбираться работники, не занятые совсем, или работники, количество задач которых ниже определенного значения. Далее будет рассмотрен алгоритм, позволяющий распределить уже занятых работников по задачам. «Венгерский алгоритм» решает данную упрощенную задачу, но при этом конкретной компетенции присуждает конкретного человека, создавая пару 1-к-1. В общем случае, возможно возникновение ситуации, когда персонала не будет хватать, поэтому, чтобы избежать таких вариантов предлагалось модифицировать данный алгоритм.

Другими словами, мы хотим, чтобы в рамках 1 задачи, если присутствует человек, обладающий помимо основной (компетенции, определенной венгерским алгоритмом для конкретной задачи) другой компетенцией с достаточно большим коэффициентом доверия, то такого работника в рамке одной задачи можно присвоить нескольким компетенциям.

Модификация «Венгерского алгоритма» заключается в сравнении компетенции работника, которого определил алгоритм на данную компетенцию, с компетенциями в рамках одной задачи. Если компетенция работника, назначенного алгоритмом на другую компетенцию, будет выше, то возможно назначение одного работника на несколько компетенций. Стартуя с разбиения по «венгерскому алгоритму», пытаемся назначить повторно работников на дополнительные компетенции. Такой подход позволит максимизировать качество выполняемой работы и оптимизировать количество человеческих ресурсов.

Теория массового обслуживания

В ходе эксплуатации системы возможно возникновение следующего случая. Рассмотрим ситуацию, когда ранее было распределение работников по задачам на базе их компетенций, но в систему приходят еще задачи в период, когда предыдущие задачи не были выполнены (таким образом у некоторых работников могут быть назначенные работы), и количество свободных людей недостаточно для выполнения новых задач. Для распределения работников в таких случаях планируется применять теорию массового обслуживания. В рамках этого мы допускаем, что часть компетенций у задач, пришедших в систему, могут быть закрыты свободными сотрудниками. Тогда применяем «Венгерский алгоритм» или его модификацию, но не для свободных работников, а у занятых наименьшим равным количеством задач.

Теория очередей – раздел прикладной математики, изучающий процессы, связанные с удовлетворением массового спроса на обслуживание, с учетом случайного характера спроса и обслуживания". Сюда относятся системы, предназначенные для обслуживания массового потока требований случайного характера, случайными могут быть как моменты появления требований, так и затраты времени на их обслуживание. Предполагается, что занятый работник не может обслужить компетенцию, пришедшую в систему, так как в текущий момент он работает над другой задачей.

Теория очередей характеризуется таким фактором, как время обслуживания, что в задачах с распределением работ не очень применимо. Зачастую заданы начало и конец проекта, но оценить время выполнения отдельных задач в проекте очень сложно.

Для решения данной проблемы была предложена модель:

- задача, которая пришла, извне будет считаться обслуживаемым объектом

- конкретный работник считается системой обслуживания данного объекта; при этом работника будем рассматривать как одноканальную систему массового обслуживания с отказами (СМО).

Идея в представлении работника в виде одноканальной системы исходит из того, что человек качественно может работать только в рамках 1 задачи, и отказ в данном случае подразумевается, как невыполнение в срок поставленной задачи.

Согласно теории, в рамках одноканальной системы массового обслуживания можно выделить вероятность то, что данный работник сейчас свободен (p0 показывает вероятность что система сейчас свободна в рамках СМО) и то, что он будет занят (p1 показывает вероятность, что система сейчас занята в рамках СМО).

$$p1 = \frac{\lambda}{\lambda + \mu}$$

$p0 = \frac{\mu}{\lambda + \mu}$, где μ – интенсивность потока обслуживания,

λ – интенсивность потока заявок

Чем выше значение $p0$, тем более предпочтительный работник. Под интенсивностью потока заявок мы будет понимать количество дополнительных работ, в которых задействован человек, а под обслуживанием – над сколькими задачами может работать человек (в данном случае равна 1).

Такой подход не позволяет оценить вероятности и понять, как они связаны со значением компетенции, а именно они являются определяющими. Для решения данной проблемы была предложена величина, аналогичная абсолютной пропускной способности, но сделан акцент на значение компетенции:

$$Q = \frac{\mu}{\lambda + \mu}$$

$A = \lambda Q$, где Q – относительная пропускная способность,

A – абсолютная пропусная способность

Предлагается ввести абсолютное значение компетенции по аналогии с абсолютной пропускной способностью.

$K = kQ$, где K – обсолютное значение компетенции,

k – значение компетенции работника

Такой подход позволил сравнивать значение с компетенциями других не занятых работников и предлагать оптимального работника. Получается, что человек, уже выполняющий задачу получает штраф к своему значению компетенции.

Данный подход вполне неплохо работает, но плохо ложится на теорию СМО и одноканальных СМО с отказами.

Как альтернативу первому способу был предложен вариант:

$$\mu = \frac{k}{\text{колиичество задач выполняемый человеком}}$$

В основу метода легла идея: чем больше значение компетенции, тем больше задач одновременно может решать человек.

Такой подход не позволяет сравнивать занятых людей со свободными, но всегда пытается сначала распределить незанятых людей, а потом уже назначать занятым дополнительные задачи.

Поиск семантически близких слов

Бывают случаи, когда из предложенных работников у нас нет человека, обладающего необходимой компетенцией. При таких ситуациях стоит либо дополнить штат сотрудников

новыми рабочими или же выбрать компетенцию из уже имеющихся, как наиболее подходящую для замены. Для этого необходимо понять семантику наименования компетенции, устранить семантическую неоднозначность, определить семантическую связь с другими словами.

Семантическое свойство слова – любая отличительная характеристика значения слова, которая служит для различения его значения от значений других слов.

Устранение семантической неоднозначности – это процесс выбора определенного смысла слов, исходя из их контекста. В этой задаче важным моментом является определение связанности разных смыслов, поскольку, хотя смысл слова выбирается из некоторого определенного множества, избранное значение слова должно наиболее соответствовать (в семантическом смысле).

В данном случае планируется сравнивать компетенцию, которую нужно заменить с компетенциями, которые существуют в системе. Задача сводится к поиску семантически близкого слова из предложенной группы слов. Для этого вводятся различные меры сходства и связанности, которые используются также в таких задачах, как: определение структуры текста, аннотирование и реферирование текстов, информационный поиск, автоматическое индексирование и автоматическая коррекция ошибок в текстах.

Для сравнения слов необходимо наличие определенных словарей.

Идеографи́ческий (семанти́ческий) слова́рь – словарь, в котором статьи упорядочены не по алфавиту, а по смыслу (лексическому значению заглавного слова или фразы). Если алфавитный словарь служит для того, чтобы узнать что-то о данном слове, то идеографический словарь служит для того, чтобы узнать что-то о данном смысле – например, какими словами можно выразить данное значение.

Некоторым приближением к идеографическому словарю являются также словари синонимов и антонимов.

Создание идеографического словаря очень затратное, требующее большого объема данных со значениями слов и их синонимами, упорядоченными в иерархические деревья.

Предлагается использование Wikidata. Викиданные (Wikidata) – это свободная, совместно наполняемая, многоязычная, вторичная база знаний, в которой собрана структурированная информация для обеспечения поддержки Википедии, Викисклада, а также других вики-проектов движения Викимедиа – по всему миру[7].

База состоит в основном из элементов, каждый из которых имеет метку, описание и любое количество синонимов. Элементы имеют уникальный идентификатор с префиксом Q и числовой частью, как, например, Douglas Adams (Q42).[8]

Утверждения детально описывают каждый элемент. Каждое утверждение состоит из свойства и его значения. Свойства в Викиданных идентифицируются кодом, имеющим префикс P и числовую часть; например, educated at (P69).[9] Для людей можно добавить свойство, указывающее на учебное заведение, значением которого будет ссылка на элемент школы. Для зданий вы можете назначить свойства географических координат с указанием долготы и широты. Свойства могут содержать ссылки на внешние базы данных и/или уточняющие свойства (квалификаторы). Элементы вместе с утверждениями образуют универсальную онтологию человеческих знаний.

Вся эта информация может отображаться на любом языке, даже когда исходные данные были введены на другом языке.

Таким образом мы получаем достаточно большой синсет на разных языках. Синсет – объединение слова, обозначающих одно понятие, со значениями других слов (синонимов), чьи лексические значения вместе формируют лексическое значение самого слова. Синсеты связаны между собой различными семантическими отношениями (гипонимия, антонимия, «часть-целое» и т. д.).

Условно меры близости слова можно разделить на 2 наиболее часто использующихся: основанные на путях (path based) и основанные на описаниях (gloss based)[10]. Первые

используют кратчайшие пути между концептами в базе знаний, а основанные на описаниях используют словарные описания концептов.

Основанные на путях меры были разработаны в основном только для IS-A отношений – гипо- и гипернимии (конкретизация и абстрагирование). То есть эти меры определены на таксономии понятий. В свою очередь таксономии имеют недостаток: один таксономический шаг (таксономическая связь) может быть более мелким, а другой наоборот – более широким.

Простейшей основанной на путях мерой является мера – кратчайшая длина дуги между 2 словами. Согласно этому подходу, мерой семантической схожести между двумя концептами является обратное значение длины кратчайшего пути в таксономии между этими концептами.

Основанная на путях мера. При этом подходе мера сходства двух концептов определяется как отношение кратчайшего пути в IS-A иерархии к диаметру таксономии. В дополнение к IS-A отношениям в путях допускаются отношения типа «часть-целое» (меронимия и голонимия).

Существует еще несколько подходов к определению семантической связанности концептов на основе их словарных описаний. Суть данного подхода довольно простая – семантическая связь двух концептов прямо пропорциональна количеству слов (или токенов), входящих одновременно в описание первого и второго концепта. Эта мера является мерой семантической связанности и может быть легко использована для различных частей речи и их комбинаций, в отличие от предыдущих мер, воспринимающих только существительные из-за использования в их определении IS-A иерархии.

В текущей системе было решено использовать меру, основанную на пути, так как она наиболее простая из предложенных.

Данный способ применяется, когда в системе нет необходимой компетенции и производится попытка заменить слово на ближайшее.

Ставится необходимое слово, для которого нужно найти его идентификатор в системе Wikidata. После этого проводится запрос с поиском “подкласс от” и “часть от” для данного слова. Данные шаблоны расположены в отдельных файлах и при необходимости могут быть изменены.

В результате запроса к Wikidata мы ищем названия компетенций, которые есть в системе. Такое проделывается для каждого слова, которого нет в системе. После к такой задаче применяются алгоритмы, описанные выше. Если не получилось заменить слово, то работники распределяются без учета описанной компетенции, а в результирующем JSON-файле отправляется неучтенная компетенция.

Заключение

Была создано микросервисное приложение, учитывающее 3 варианта сценариев, возникающих при назначении работников на задачи в научном проекте.

Описанные в этой статье методы работают в рамках некоторых ограничений. Для первого метода необходимо, чтобы количество работников, которых мы разбиваем, был равен суммарному количеству компетенций для пришедших в систему задач. Поиск семантически близких слов находит компетенции, связанные с естественными науками достаточно хорошо, но в других случаях поиск может быть затруднен. Для решения данной проблемы можно создавать в процессе работы программы свои словари, что и послужит дальнейшим улучшением системы.

Библиографический список

1. *Сутягин* А.Н. Организация научных исследований. // Конспект лекций. – Рыбинск, 2015 г.
2. Новиков Д.А., Суханов А.Л. Модели и механизмы управления научными проектами в вузах. // Институт управления образованием РАО. – Москва, 2005 г.
3. Баранов В.В. Исследование систем управления/ Баранов В.В., Зайцев А.В., Соколов С.Н.- Издательство: Альпина Паблишер, 2013 – 216с.

4. Бурков В. Н., Новиков Д. А. Как управлять проектами. М.: Синтег, 1997. – 188 с.
5. Матвеев А.А., Новиков Д.А., Цветков А.В. Модели и методы управления портфелями проектов. М.: ПМСОФТ, 2005. – 206 с.
6. Задача о назначениях [Электронный ресурс] URL: https://habr.com/ru/post/63982/ (дата обращения 11.01.2021)
7. Официальный сайт Викиданных [Электронный ресурс] URL: https://www.wikidata.org/wiki/Wikidata:Introduction/ru (дата обращения 21.03.2021)
8. Denny Vrandečić. 2012. Wikidata: a new platform for collaborative data collection. In Proceedings of the 21st International Conference on World Wide Web (WWW '12 Companion). Association for Computing Machinery, New York, NY, USA, 1063–1064. DOI:10.1145/2187980.2188242
9. John Samuel. (2018). Towards Understanding and Improving Multilingual Collaborative Ontology Development in Wikidata. Presented at the Wiki Workshop 2018 (held at The Web Conference 2018), Lyon, France: Zenodo. DOI: 10.5281/zenodo.1219240
10. Анисимов А.В., Лиман К.С., Марченко А.А. Методы вычисления мер семантической близости слов естественного языка // Киевский национальный университет им. Т. Шевченко. – Киев, 2010 г.

SYSTEM OF OFFERING EMPLOYEES FOR JOBS FOR SCIENTIFIC PROJECTS

Nikulin Maxim V., Chupin Anton V.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, ivanov@email.ru

In the modern scientific world, research and coherent work is an integral part of scientific activity. The main task of this work is to create a system for proposing an employee for a position for such work and research. These works can be organized by using project management systems, but scientific projects have their own characteristics. This article discusses research projects and their key features. For scientific projects, the concept of a job proposal system was proposed, taking into account their characteristics. During the study of projects, it was determined that all situations for the appointment of workers can be divided into three groups: whether there is an employee suitable for the task and is currently free, there is an employee suitable for the task but who is busy at the moment, and there is no suitable employee for the task. We used the assignment problem, queuing theory and search for semantically close words, respectively, for each of the possible options. The abilities of people are known in advance.

Key words: Scientific project, management, employee proposals for a position, application software.

УДК 004.434:004.94+004.052.4

АВТОМАТИЗАЦИЯ ПРОВЕРКИ UCD СТУДЕНТОВ

Отинов Андрей Валерьевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, otinovandry@gmail.com

Дацун Наталья Николаевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, nndatsun@inbox.ru

Унифицированный язык графического моделирования UML является действующим стандартом нотации (ISO/IEC 19505:2012) для визуализации моделей при разработке программного обеспечения. UML предоставляет основные рекомендации и правила для визуализации и понимания сложных программных систем. Именно по этой причине он стал частью учебных программ для курсов программной инженерии во многих университетах мира. Однако, многие студенты, преподаватели и разработчики программного обеспечения допускают ошибки при построении или проверке диаграмм. Затрачиваемое время на проведение верификации диаграмм – важный ресурс, который возможно сократить путём автоматизации. В этой работе описывается список наиболее распространённых ошибок студентов в диаграммах вариантов использования. Представлен результат проектирования, реализации и тестирования программного средства для верификации выделенных ошибок.

Ключевые слова: UML, прецедент, ошибки студентов, автоматизация верификации.

Язык графического описания для объектного моделирования в области разработки программного обеспечения (ПО) Unified Modeling Language (UML) [1], [2] был представлен в октябре 1996 года. С того момента язык UML закрепил за собой звание стандарта визуализации системного проектирования. Данный стандарт стал частью учебных программ множества университетов по всему миру.

Но такая распространённость UML не упрощает задачу его изучения. Было проведено исследование, в результатах которого значилось то, что множество разнообразных пользователей данного стандарта склонны совершать однотипные ошибки при создании своих UML-моделей [3].

Также, если рассматривать процесс верификации студенческих диаграмм преподавателем, то он требует значительного промежутка времени, особенно если это касается проверки десятков работ у целой группы студентов. Поэтому автоматизация этого процесса становится актуальной и важной задачей.

В данной работе рассмотрена верификация диаграмм прецедентов (UseCase Diagram, UCD) в рамках проекта системы автоматизации верификации UML диаграмм – UCD, AD (диаграмм активностей) и CD (диаграмм классов). Количество ошибок в диаграммах этого типа является одним из наибольших, поскольку данный тип диаграмм является первым этапом создания моделей систем при объектно-ориентированном подходе. И именно в самом начале изучения языка UML чаще всего допускаются ошибки.

Для верификации возможных ошибок в UCD было проведено исследование около 140 студенческих работ студентов ПГНИУ, в рамках которого был выделен список наиболее распространённых ошибок студентов. Также было проведено изучение научных публикаций, связанных с данной тематикой [3], [4], [5].

Список всех идентифицированных ошибок был классифицирован по трём группам, соответствующим роду ошибки: лексические, синтаксические и семантические. Классифицированный список ошибок с примерами представлен в таблице ниже.

Таблица. Возможные ошибки, возникающие при создании UC диаграмм

Тип ошибки	Описание
Лексическая	Отсутствует граница системы
Лексическая	Некорректное имя актора: должно быть представлено именем существительным, начинаться с большой буквы
Лексическая	Некорректное название прецедента: должно быть представлено в виде действия (глаголом или именем существительным отглагольной формы), начинаться с большой буквы
Лексическая	Отсутствует название системы (имя субъекта)
Лексическая	Использование элементов, не входящих в список, разрешенных стандартом UML для UCD
Лексическая	Прецеденты должны быть представлены соответствующим элементом
Синтаксическая	Отсутствует точка расширения у прецедента с отношением расширения
Синтаксическая	Отсутствует текст в точке расширения прецедента
Синтаксическая	Отсутствует текст в условии расширения
Синтаксическая	Имена акторов должны быть уникальными
Синтаксическая	Имена прецедентов должны быть уникальными
Синтаксическая	Использование отношения включения между акторами
Синтаксическая	Использование отношения расширения между акторами
Синтаксическая	Нахождение актора внутри системы
Синтаксическая	Использование отношения ассоциации между прецедентами
Синтаксическая	Прецедент не имеет отношений с другими элементами диаграммы. Прецедент должен иметь отношение с актором в виде ассоциации, либо иметь отношения расширения, дополнения или включения с другими прецедентами
Синтаксическая	Использование отношения обобщения не между двумя акторами или прецедентами
Синтаксическая	Использование отношения расширения не между двумя прецедентами
Синтаксическая	Использование отношения включения не между двумя прецедентами
Синтаксическая	Актор не имеет ни одного отношения ассоциации с прецедентами
Семантическая	Родительский прецедент, имеющий отношения включения или расширения, не имеет отношения ассоциации
Семантическая	Прецедент с отношением включения, включает всего один прецедент
Семантическая	Два и более акторов с одинаковыми наборами ассоциированных прецедентов
Семантическая	Отношения включения направлены не в ту сторону
Семантическая	Отношения расширения направлены не в ту сторону
Семантическая	Злоупотребление отношением включения
Семантическая	Неиерархическая структура прецедентов

На основании полученного списка ошибок становится возможным определить методологию верификации данных ошибок.

На текущий момент существует два основных подхода в верификации диаграмм вариантов использования: Object-Oriented Reading Techniques (OORT) [6] – объектно-ориентированные техники чтения и Checklist-Based Reading (CBR) [7] – чтение, основанное на списке требований.

Метод OORT – это методология, предполагающая проведение серии различных чтений диаграммы и сопоставления её с уже существующей. Но данный метод предполагает

человеческое участие и основан на различиях и особенностях человеческого восприятия информации, вследствие чего не может быть автоматизирован.

Метод CBR представляет собой метод чтения на основе контрольных списков, который заранее составляется путем анализа предметной области и учетом всех необходимых требований. В текущей работе данный метод может быть реализован на основе контрольного списка наиболее распространённых ошибок, что является его достоинством. Также достоинством данного метода можно считать то, что данная технология имеет возможность расширения, при выявлении новых, не фиксированных ранее случаев совершения ошибок. К недостаткам метода можно отнести то, что полнота покрытия всех возможных случаев совершения ошибок зависит от полноты контрольного списка. В системе верификации UCD, представленной в данной работе, выбрана методология CBR.

При формировании требований к системе верификации диаграмм UML были проанализированы варианты представления входных данных. Поскольку UML является языком графического моделирования, автоматизация верификации модели в ее графическом представлении с помощью программного средства является невозможной. Становится необходимым формализовать входные данные. Эта задача становится актуальной в связи с развитием поискового подхода к проектированию архитектуры программного обеспечения [8].

Если же перевести описание моделей UML в формальную форму, это значительно упростит их анализ. На данный момент существует несколько направлений формального описания диаграмм UML: сети Петри [9], трансформации на основе графов [10], операционная семантика [11], темпоральная логика [12], конечные автоматы [13], денотационная семантика [14].

Современные среды графического моделирования диаграмм UML применяется формат XML Metadata Interchange (XMI) [15]. Этот формат хранения метаданных был разработан Object Management Group и является открытым для общественного использования. XMI предназначен для хранения и передачи UML-моделей между приложениями. Хоть данный формат и предполагает определённую степень верификации хранимых данных, он не гарантирует полноценное отсутствие синтаксических и семантических ограничений. В нашей системе верификации UCD для файлов входных данных моделей выбран XMI формат.

Следующим шагом нашего исследования стал выбор средств моделирования (инструмента создания) диаграмм вариантов использования, поскольку это ПО влияет на возможность построения автоматизированной системы верификации диаграмм, а также на качество проводимого этой системой анализа. Был проведён анализ существующих программных средств моделирования, в результате которого были выделены следующие критерии, перечисленные в порядке убывания степени их значимости для нашей системы:

1) Средство моделирования диаграмм UML должно удовлетворять всем требованиям стандарта языка UML для построения диаграмм. В частности, данная система обязана представлять использования всех задекларированных элементов, используемых при построении диаграмм вариантов использования.

2) Средство моделирования должно иметь возможность экспортировать спроектированную модель в формате XMI.

3) Экспортируемый файл для хранения данных о спроектированной модели должен соответствовать стандартам XMI, а также содержать информацию обо всех видах отношений между компонентами модели, их координаты, используемые при графическом отображении. Отношения между элементами должны быть однозначны и стандартизированы. Без соблюдения данного пункта дальнейший анализ, проектирование и реализация системы автоматизации верификации диаграмм становятся неосуществимыми в силу очевидной невозможности определения сущностей и их взаимодействия друг с другом.

4) Средство моделирования должно предоставляться по открытой лицензии в свободное некоммерческое использование. Это условие является необходимым, поскольку потенциальная группа пользователей системы верификации – это студенты и преподаватели.

5) Средство моделирования должно обладать простотой установки и дальнейшего использования потенциальным пользователем, а также быть доступным не только для уверенных пользователей ПК.

6) Средство моделирования должно обладать дружественным интерфейсом.

В результате проведенного исследования на основании описанных критериев был сделан выбор в пользу средства моделирования GenMyModel [16].

Для анализа бизнес-процессов при верификации UML диаграмм была разработана диаграмма по методологии IDEF0 (см. рис. 1). Данная диаграмма демонстрирует тот путь, который проходят данные, с момента создания пользователем модели с помощью средства графического моделирования GenMyModel до момента её верификации и экспорта результата верификации в текстовый документ.

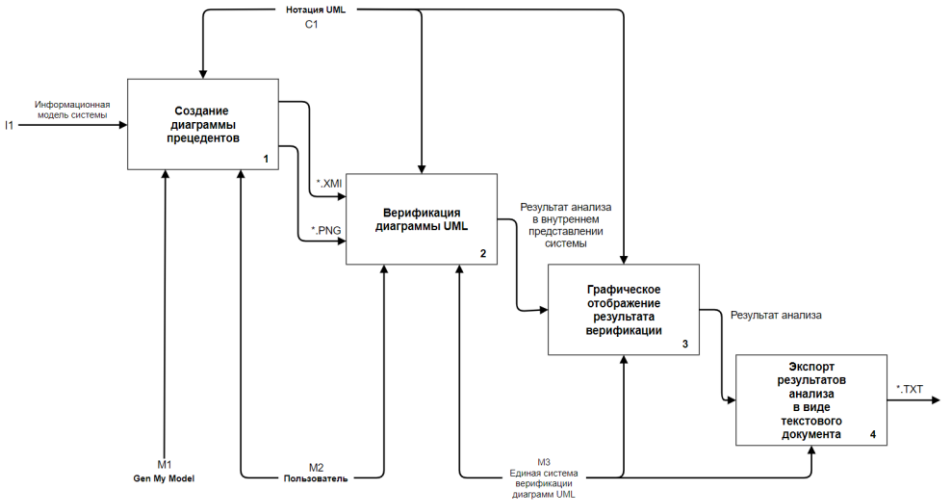


Рис. 1 Диаграмма IDEF0 системы автоматизации верификации UML диаграмм

На рис. 2 продемонстрирована архитектура системы автоматизации верификации UML диаграмм на этапе проектирования в виде диаграммы классов.

Для хранения информации о считываемых элементах диаграммы прецедентов были определены классы «Element» и «Arrow», представленные на рис. 3. Базовым классом является класс «Element». Он имеет атрибуты, общие для большинства видов элементов UCD. Класс «Arrow», производный от класса «Element», определён для хранения элементов с типом отношений между двумя элементами: включение, расширение, обобщение, точка расширения, ассоциация.

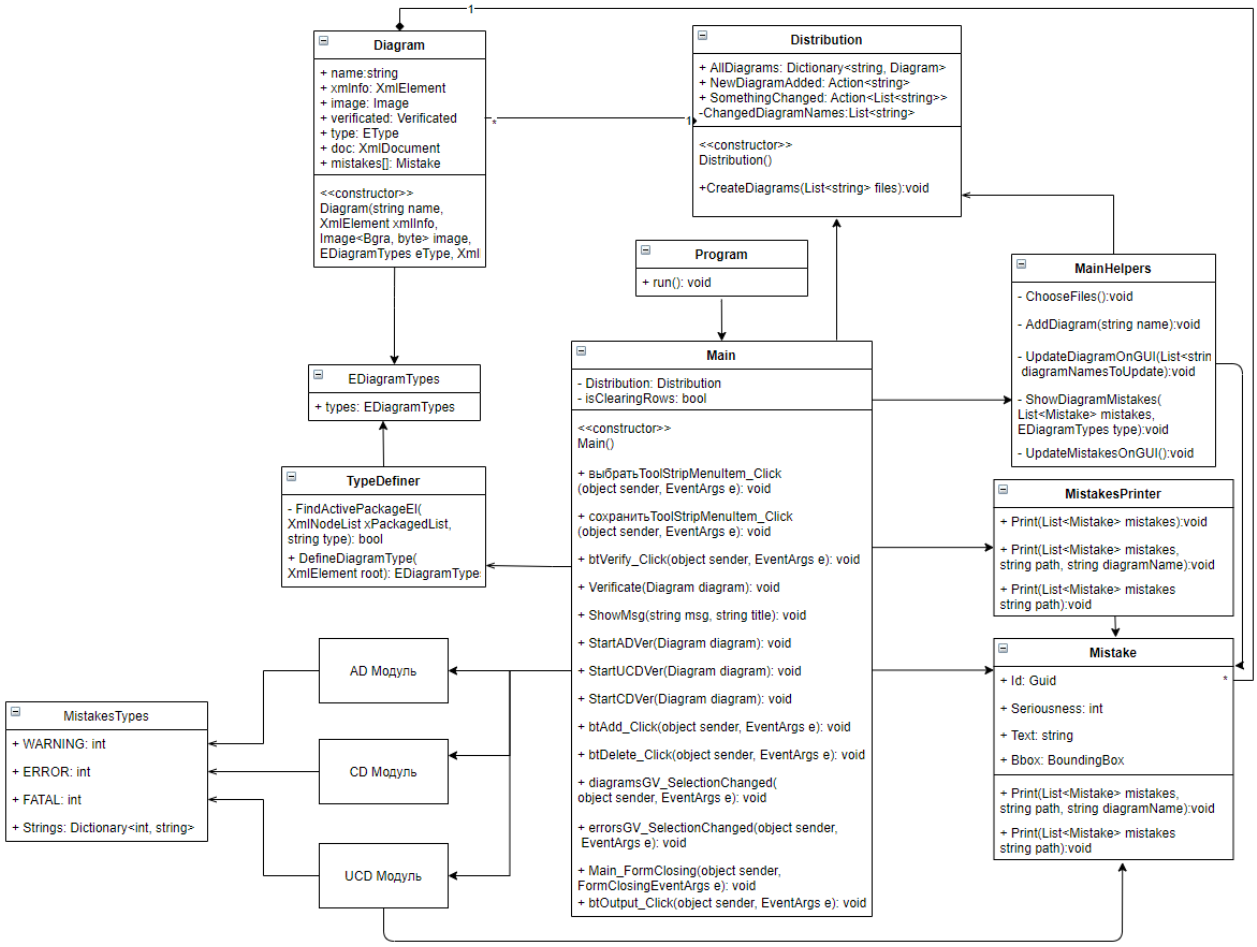


Рис. 2 Диаграмма классов системы автоматизации верификации UML диаграмм

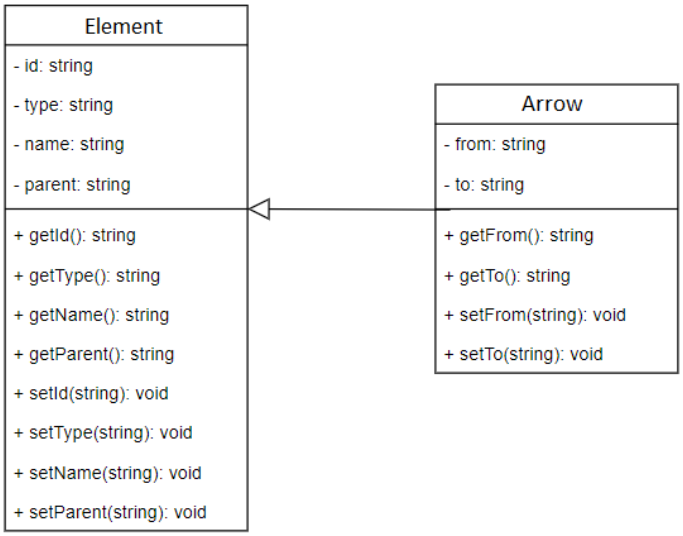


Рис. 3 Классы «Element» и «Arrow», используемые при автоматизации проверки UCD

На рис. 4 продемонстрированы компоненты разработанного модуля верификации диаграмм вариантов использования.

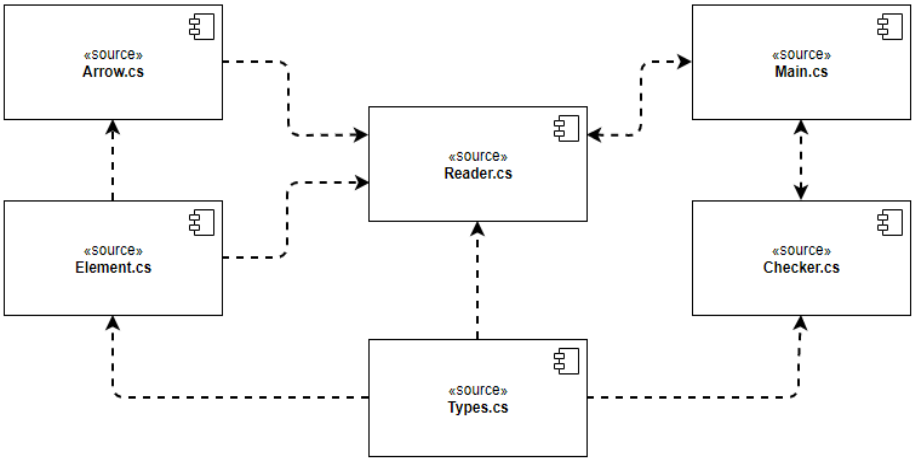


Рис. 4 Диаграмма компонентов модуля верификации UCD

Для разработки системы автоматизации верификации UML диаграмм был выбран объектно-ориентированный язык программирования C# с интерфейсом Windows Forms.

Тестирование корректности работы системы автоматизации верификации UML диаграмм проводилось на работах студентов ПГНИУ. В тестировании модуля верификации UCD использованы UCD из около 140 работ студентов, которые содержали различные ошибки. Наиболее распространённым типом ошибок являются лексические ошибки. Процентное соотношение успешно обнаруженных модулем верификации UCD ошибок к реальным ошибкам из работ студентов представлено на рис. 5.

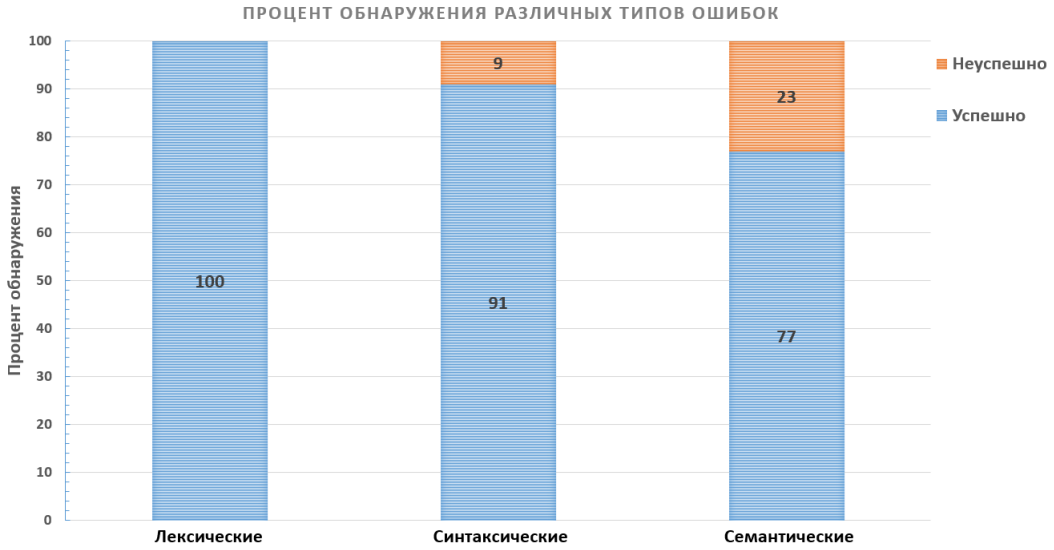


Рис. 5 Процент обнаружение ошибок в работах студентов модулем верификации UCD

Таким образом, на основе UCD, подготовленных студентами ПГНИУ, был сформирован список наиболее распространённых ошибок, которые допускаются при построении диаграмм прецедентов. Этот список использован как контрольный в методе CBR, примененном при разработке модуля верификации UCD. Обоснован выбор формата данных моделей на языке UML и средства моделирования для создания этих моделей, которые поступают на вход системе автоматизации верификации UML диаграмм. На основе анализа бизнес-процессов при верификации UML диаграмм спроектирована система автоматизации верификации UML диаграмм, в рамках которой реализован и протестирован модуль верификации UCD. Степень обнаружения ошибок модулем высокая, что свидетельствует о достижении цели нашего исследования.

Библиографический список

1. *Booch G., Rumbaugh J., Jacobson I.* The Unified Modeling Language User Guide. Addison-Wesley, Addison-Wesley Professional, 2005.
2. *Rumbaugh J., Jacobson I., Booch G.* Unified Modeling Language Reference Manual, Addison-Wesley Professional, 2004.
3. *Mistakes* in UML Diagrams: Analysis of Student Projects in a Software Engineering Course / S. Chren [et al.] // Proc. IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training. 2019. Pp. 100–109.
4. *Chytalová K.* Katalog chyb v UML diagramech [Электронный ресурс] URL: https://is.muni.cz/th/dqtso/Katalog_castych_chyb_v_UML_diagramech.pdf (дата обращения: 18.05.2021).
5. *Jennifer C.* Verification and Validation UML models [Электронный ресурс] URL: https://docplayer.net/24240668-Uml-models-lecture-10-part-1-verification-and-validation-uml-models-2-non-uml-models-verification-and-validation.html (дата обращения: 18.05.2021).
6. *Object-Oriented* Reading Techniques for Inspection of UML Models – An Industrial Experiment / R. Conradi [et al.] // Lecture Notes in Computer Science. 2003. Vol. 2743. Pp. 483-500.
7. *Kösters G., Six H., Winter M.* Validation and Verification of Use Cases and Class Models // Proc. 7th International Workshop on Requirements Engineering: Foundations for Software Quality. 2001. Pp. 1-10.
8. *Никульчев Е.В., Плужник Е.В., Лукьянчиков О.И.* Проектирование распределенных информационных систем обработки больших объемов данных в гибридной облачной инфраструктуре // Вестник РГРТУ. 2014. № 4. Ч. 1. С. 135-138.
9. *Baresi L., Pezzè M.* On formalizing UML with High-Level Petri Nets // Lecture Notes in Computer Science. 2001. Vol. 2001. Pp. 276-304.
10. *Use* Case Analysis Based on Formal Methods: An Empirical Study / Oliveira M. Jr. [et al.] // Lecture Notes in Computer Science. 2015. Vol. 9463. Pp. 110-130.
11. *Dynamic* Meta Modeling: A Graphical Approach to the Operational Semantics of Behavioral Diagrams in UML / G. Engels [et al.] // Lecture Notes in Computer Science. 2000. Vol. 1939. Pp. 323-337.
12. *Klimek R., Szwed P.* Formal Analysis of Use Case Diagrams // Computer Science. 2010. Vol. 11. Pp. 115-131.
13. *Analysing* UML Active Classes and Associated State Machines – A Lightweight Formal Approach / G. Reggio [et al.] // Lecture Notes in Computer Science. 2000. Vol. 1783. Pp. 127-146.
14. *Evans A., Kent S.* Core Meta-Modelling Semantics of UML: The pUML Approach // Lecture Notes in Computer Science. 1999. Vol. 1723. Pp. 140-155.
15. *XML* Metadata Interchange (XMI) Specification [Электронный ресурс] URL: https://www.omg.org/spec/XMI/2.5.1/PDF (дата обращения: 18.05.2021).
16. *Сайт* продукта GenMyModel [Электронный ресурс] URL: https://www.genmymodel.com/ (дата обращения: 18.05.2021).

AUTOMATION OF VERIFICATION UCD DIAGRAMS OF STUDENTS

Otinov Andrei V.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, otinovandry@gmail.com

Datsun Natalia N.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, nndatsun@inbox.ru

Abstract. Unified Modeling Language (UML) is the current notation standard (ISO/IEC 19505:2012) to visualize models in software development. UML provides essential guidelines and rules to visualize and understand complex software systems. This is the reason why it has become part of curricula for software engineering courses at many universities worldwide. However, many students, teachers or software developers make mistakes when constructing or miss these on checking the correctness of these diagrams. The time spent on verification of diagrams is an important resource that can be reduced by automating. This paper describes a list of the most common students mistakes in use case diagrams. The design, implementation, and testing of verification module of identified mistakes were presents.

Keywords: UML, use case, mistakes students, verification automation.

УДК 004.89;004.416.3

РАЗРАБОТКА ВИЗУАЛЬНЫХ РЕДАКТОРОВ ОНТОЛОГИЙ НА ПРИНЦИПАХ АДАПТИВНОСТИ НА ПРИМЕРЕ РЕДАКТОРА ONTSPACE

Шарцев Григорий Константинович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, me@shgk.me

В статье с использованием метафоры солнечной системы описывается подход к разработке адаптируемых визуальных редакторов онтологий на примере веб-редактора OntSpace. В основу концепции положен управляемый моделями метод проектирования и разработки интеллектуальных систем. В качестве модели выступает состоящая из трех уровней метамодель самого редактора онтологий. На уровне ядра метамодели описываются знания о редакторе онтологий и о метамодели. На уровне профилей определяется работа редактора для решения конкретного типа задачи. На уровне плагинов описывается работа редактора для решения конкретной подзадачи. Все уровни метамодели представлены в виде онтологий. Высокоуровневый интерфейс для выбора профиля и подключения плагинов позволяют легко адаптировать редактор к предпочтениям различных категорий пользователей и специфике предметной области. Работа с метамоделью в формате онтологии позволяет разработчикам, используя визуальные средства графического редактора, легко адаптировать визуальные средства самого редактора и его функциональные возможности под решаемую задачу без необходимости внесения изменений в исходный программный код. Описываются основные компоненты редактора OntSpace и механизм их встраивания в сторонние приложения.

Ключевые слова: редактор онтологий, онтология, метамодель, мета-онтология, адаптируемый визуальный редактор онтологий, онтологически-управляемые приложения

Введение

Применение онтологий в ИТ-сфере и их внедрение в процесс разработки приложений в широком спектре различных областей и направлений продолжает расти [1]. Большую популярность набирает подход, основанный на OBDA-парадигме (Ontology-Based Data Access [2, 3]), в рамках которой управление данными основано на использовании онтологий. Онтологии активно применяются в геоинформатике, биомедицине, экологии и в других областях для разработки адаптируемых к потребностям пользователей сред для решения широкого круга задач, в частности, для создания персонифицированных рекомендательных систем, для автоматического анализа текстов на естественных языках безотносительно к тематической направленности текста и стилю публикации и во многих других задачах. Растёт потребность в разработке онтологически управляемых решений и, соответственно, в разработке онтологий и инструментальных средств их создания, анализа и интеграции.

В настоящее время существует большое число визуальных редакторов онтологий, однако даже самые популярные из них, например Protégé [4], не соответствуют современным требованиям по адаптации визуального редактора к специфике решаемой задачи и персональным предпочтениям обычных пользователей, а ориентированы на инженеров-онтологов [5]. Отсутствие механизмов такой адаптации усложняет разработку онтологий и онтологически управляемых решений, т.к. предполагает, что эксперты предметной области и разработчики онтологически управляемых приложений должны работать в паре с профессиональным инженером по знаниям (онтологическим инженером) для совместной разработки онтологии в среде профессионального редактора.

Наша цель – изменить роль онтологического инженера в процессе создания онтологически управляемых приложений, предоставив возможность эксперту самостоятельно разрабатывать онтологии своих предметных областей. При этом у онтологического инженера появляется возможность сосредоточиться на анализе построенных экспертом онтологий и при помощи высокоуровневых средств трансформировать всю или часть созданной экспертом онтологии, устранив типичные ошибки. Таким образом, за инженером-онтологом остаются функции контроля и проверки качества онтологии, но при этом не требуется больших временных затрат на обязательную совместную работу. Мы считаем, что эта цель может быть достигнута путём разработки визуального редактора онтологий, имеющего высокоуровневые средства для адаптации к потребностям различных категорий пользователей (экспертов, инженеров-онтологов, разработчиков онтологически управляемых решений) и различным типам задач.

Основная идея предлагаемого подхода

Наш подход предполагает, что адаптация визуального редактора онтологий к специфике предметной области решаемых задач и персональным предпочтениям пользователей также должна выполняться на принципах онтологического инжиниринга. В [5, 6] описан прототип визуального редактора онтологий ОНТОЛИС, разрабатываемый в Пермском государственном национально исследовательском университете. Работа этого редактора управляется так называемой мета-онтологией (онтологией, содержащей знания о редакторе онтологий и о метапонятиях предметных онтологий).

Развивая заложенную в ОНТОЛИС идею подхода к разработке онтологически управляемых редакторов онтологий, мы предлагаем всю основную функциональность редактора, включая правила трансформации и семантическую фильтрацию, оформить в виде управляемой на мета-уровне системы плагинов. Хотя многие существующие редакторы и имеют развитую систему плагинов, позволяющую добавлять необходимую функциональность по требованию, такие редакторы остаются сложными в использовании для неопытных пользователей. Другие редакторы, позволяющие эксперту достаточно легко создавать предметные онтологии, зачастую мало функциональны и не подходят профессиональным пользователям [8]. Кроме того, сама по себе система плагинов не решает указанную проблему полностью, так как при расширении системы плагинов требуется

привлечение программистов к разработке и встраиванию новых плагинов в уже существующую систему.

Для того чтобы учесть требования как экспертов (обычных пользователей), так и онтологических инженеров, мы предлагаем усовершенствовать управляемые мета-онтологией механизмы настольного редактора ОНТОЛИС и разработать новый веб-редактор онтологий OntSpace с развитым механизмом адаптации под персональные предпочтения различных категорий пользователей. Механизм адаптации должен учитывать как поддержку различных, ориентированных на персональные предпочтения пользователей, способов визуализации вершин и дуг онтологии, так и возможность переопределения реакции редактора на типичные действия различных пользователей, а также переопределения способов семантической фильтрации и интерпретации взаимосвязанных вершин онтологии с учётом их контекста. При этом редактор должен оставаться максимально доступным для обычного пользователя и предоставлять удобные инструменты совместной групповой работы пользователей различных категорий.

Важное значение имеют вопросы программного встраивания редактора онтологии в сторонние онтологически управляемые приложения. В простейшем случае встраивание реализуется посредством экспорта и импорта онтологии в поддерживаемом формате. Но такого рода механизм встраивания усложняет разработку самого онтологического решения, так как расширение и изменение онтологии выполняется в среде стороннего редактора. При разработке редактора важно изначально прорабатывать проектные решения по способам его встраивания в другие приложения.

Отметим также, что создание в среде визуальных редакторов онтологий с аксиоматикой на практике является слишком сложным процессом для непрофессионального пользователя. Мы считаем, что для многих задач в разработке онтологически управляемых решений достаточно легковесных онтологий с заранее определённой ограниченной аксиоматикой на базе основных парадигматических типов связей («класс-подкласс», «часть-целое» и др.). Это позволит не усложнять редактор и сделать его доступным для обычных пользователей, не теряя при этом базовую аксиоматику онтологий.

OntSpace – редактор с космической моделью

Основа онтологически управляемого редактора – модель его мета-уровня (метамодель), содержащая знания о самом редакторе онтологий. Концепция мета-уровня в редакторе ОНТОЛИС основана на сборочном принципе. Мета-уровень описывает, какие библиотеки и как подключаются к редактору, их параметры, способ подключения, а также некоторую дополнительную метаинформацию. При этом интерпретация вершин и дуг онтологии остаётся в процедурной части редактора.

В редакторе OntSpace определен оптимальный набор базовых понятий и связей, через который можно описать все остальные понятия и связи, используемые на мета-уровне. Для наглядности концепцию мета-уровня OntSpace предлагается описать его в терминах компонентов солнечной системы (см. рис. 1).

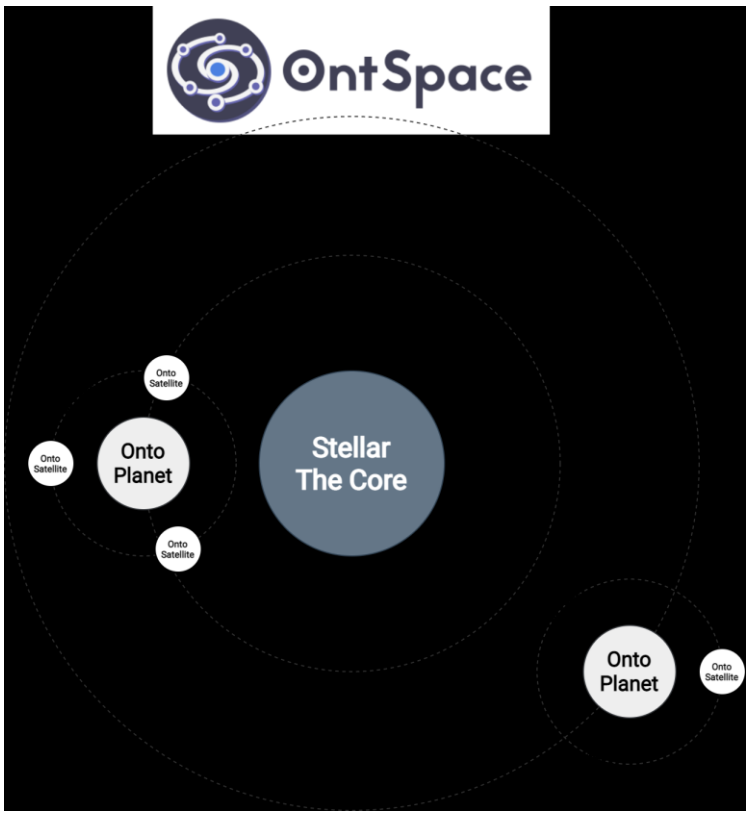


Рис. 1 Мета-уровень OntSpace как метафора солнечной системы

Центр всего редактора – это ядро мета-уровня. Данный уровень представлен мета-онтологией, которая описывает знания редактора о самом себе: знания о других уровнях системы, о всех типах сущностей, их свойствах и взаимосвязях. Этот же уровень определяет те экземпляры сущностей, которые отвечают за описание поведения редактора (его алгоритм работы) в процессе визуального редактирования пользователем своих онтологий. Такими сущностями являются: редактор и элементы его интерфейса; визуальное представление онтологии в виде графа (толщина линии дуг, фон вершины, шрифт, и т.д.); сущности Профиль и Плагин (описаны далее).

Ядро мета-уровня – это центр вселенной редактора, его Солнце. Ядро должно быть зашито в редактор, неизменяемо и едино для всех пользователей. На рисунке 2 представлен фрагмент ядра системы, описывающий основные свойства визуального представления графа онтологии. Этот фрагмент описывает концепт параметров визуализации visOptions, включающий множество различных параметров визуального представления вершин (nodes) и дуг (edges) онтологии. Для описания типов сущностей и типов их свойств используются отношения rdfs:domain и rdfs:range [7, с. 103].

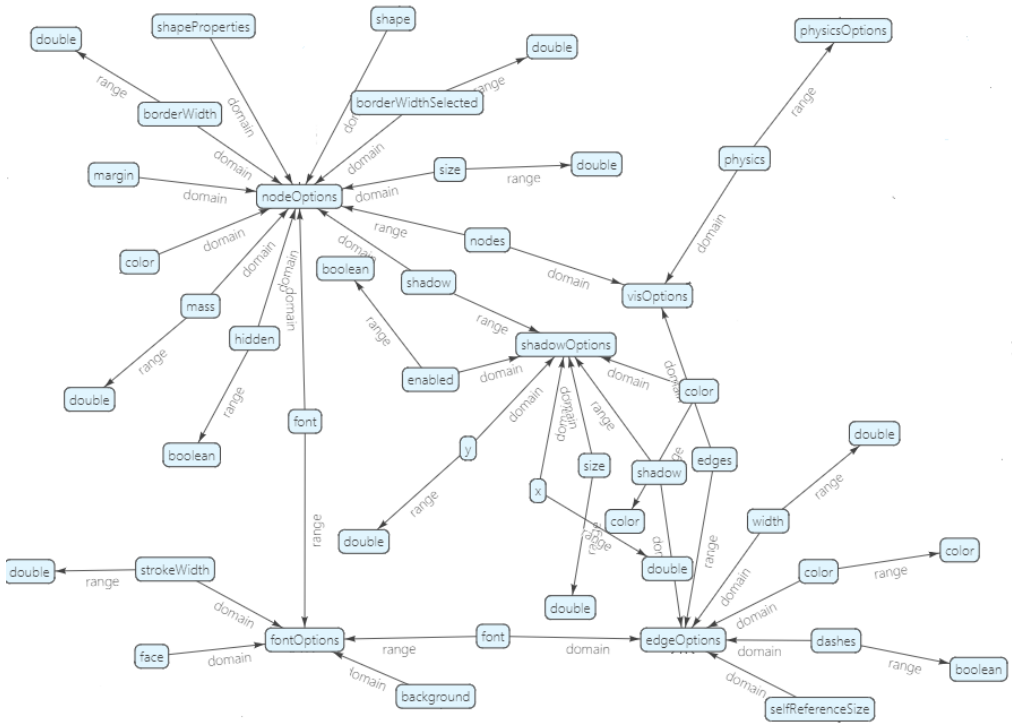


Рис. 2 Фрагмент ядра, описывающий часть опций визуализации

Следующий уровень – профили. Каждый профиль – мета-онтология, которая определяет профиль редактора, его адаптацию под решение конкретного класса задач или под потребности определённой группы пользователей. Например, профиль разработки простых таксономий, профиль для разработки онтологий в сфере IoT с определённым типовым набором связей или профиль для пользователей, привыкших к редактору ОНТОЛИС. На этом уровне определяются свойства сущностей конкретного экземпляра редактора и подключаются элементы следующего уровня – плагины. В нашей концепции профили – это планеты солнечной системы редактора.

Последний уровень – плагины, представляющие собой мета-онтологию решающих определённые подзадачи подключаемых модулей, состоящую из описания:

1) некоторого подмножества сущностей редактора;
2) действия, которое выполняется с этим подмножеством сущностей;
3) условия, при котором действие выполняется.

Такой плагин может статично определять различные аспекты визуализации онтологии (например, определять внешний вид вершин, устанавливать фон вершины изображением, хранящимся в атрибутах этой вершины и т.п.) или определять изменение свойств визуализации в динамике при наступлении определенных пользовательских событий (например, при клике на некоторую вершину выделять определенным цветом эту вершину или всю иерархию ее родительских вершин, или все вершины, связанные с ней определенными типами связей). При помощи плагинов возможно изменение интерфейса самого редактора или его интеграция с другими системами на принципах встраивания (например, добавление поля выполнения запроса к онтологии на стороннем сервере с визуализацией результатов запроса). Кроме того, плагины используются для подключения к редактору группы других плагинов.

В нашей метафоре солнечной системы плагины – это спутники планет. Какие-то спутники являются неотъемлемой частью планеты, т.е. её естественными спутниками. Другие, подобно искусственным спутникам, могут устанавливаться дополнительно и использоваться на разных планетах. Так и плагины редактора, могут быть как неотъемлемой частью профиля, так и самостоятельной сущностью, подключаемой, по мере необходимости, к любому профилю или плагину.

Пользователь редактора – это путешественник, перелетающий с планеты на планету для решения своих задач во вселенной онтологии. С помощью системы планет и личных искусственных спутников он может находить или создавать для себя желаемые условия для выполнения своих задач. Таким оразом, редактор может адаптироваться к разным задачам, а пользователь – выбирать нужный профиль и набор плагинов, которые адаптируют редактор для работы с онтологией в визуальном, интерактивном и прагматическом аспектах.

Компоненты редактора OntSpace

OntSpace – веб-приложение, и, как любое веб-приложение, является клиент-серверным. Реализация редактора как SPA (одностраничного) приложения поможет явно разделить его на клиентское приложение, предоставляющее графический интерфейс работы с редактором, и серверную часть, предоставляющую API.

Взаимодействие клиентской и серверной частей в процессе создания онтологий предлагается реализовать на базе фреймворка Logux [8], который позволяет достаточно просто разработать offline-first веб-приложение с возможностью совместной работы в режиме реального времени на принципах репликации без конфликтов с использованием CRDT (Conflict-free Replication Data Types) [9]. Это решение отлично подходит к разработке редакторов онтологий, где относительно небольшое число редактируемых типов сущностей, в основном, это поименованные вершины и дуги, а сам процесс редактирования сводится к ограниченному набору действий по их созданию, удалению и изменению значений их свойств (имен, атрибутов, координат и др.). Offline-first подход позволяет уменьшить связность клиента и сервера в реализации редактора, упрощая встраивания клиента в другие приложения.

В обобщенном виде основные компоненты визуального редактора онтологий OntSpace представлены на рисунке 3. Вместе с Logux в состав серверной части редактора входит публичное API (компонент @ontspace/server выделен на рис. 3 розовым цветом), предоставляющее программный интерфейс для взаимодействия с онтологиями: операции создания, чтения, обновления и удаления онтологий.

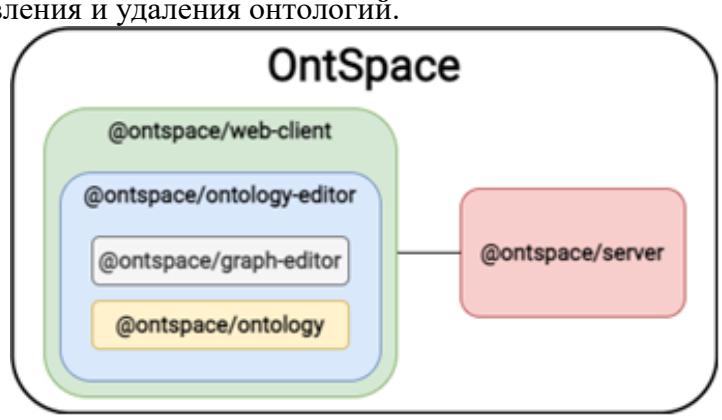


Рис. 3 Компоненты OntSpace

Компонент @ontspace/ontology представляет собой модуль по удобной работе с онтологией во внутреннем формате OntSpace. Это SDK для использования онтологии во внутреннем формате редактора. Данный компонент позволяет упростить использование в сторонних приложениях онтологий, разработанных в среде редактора OntSpace, а также экспортировать их в другие форматы, такие как OWL, RDF и ONT.

Компонент @ontspace/graph-editor реализует веб-компонент визуального редактирования графов. Такой компонент можно использовать в широком круге веб-приложений, где требуются простые функции по визуальному редактированию, просмотру и анализу онтологий в форме графа. Высокоуровневый компонент @ontspace/ontology-editor интегрирует перечисленные выше компоненты @ontspace/ontology, @ontspace/graph-editor и мета-онтологию редактора в единый визуальный редактор онтологий. Этот компонент позволяет работать с визуальным редактором графов и графическим представлением онтологий в среде других веб-приложений с возможностью его адаптации на базе мета-онтологий.

Клиентское приложение @ontspace/web-client – это веб-приложение OntSpace, предоставляющее помимо возможностей работы с онтологиями такие типовые функции веб-приложений как поиск и создание планет-профилей, спутников-плагинов; создание и сохранение онтологий в профиле пользователя; обеспечение совместной работы групп пользователей по редактированию онтологий.

Для обеспечения групповой работы пользователей редактора онтологий предлагается разделить редактируемую сущность онтологии на три части:

1) непосредственно содержательная часть онтологии, представленная множеством взаимосвязанных концептов;
2) атрибуты, привязанные к вершинам и дугам онтологии, необходимые для её визуализации, например, координаты местоположения вершин на холсте (важно отделить содержательную часть онтологии от её графического представления в редакторе);
3) рабочая область, состоящая из онтологии в совокупности с данными её отображения в редакторе в некотором фиксированном профиле с определённым набором плагинов.

Такое разделение позволяет получать различные представления одной и той же онтологии в зависимости от предпочтений различных групп пользователей.

Заключение

Описанный подход предлагает высокоуровневые средства адаптации редактора онтологий к различным задачам, а его модульная структура – интеграцию в сторонние приложения.

С одной стороны, данный подход нацелен на уменьшение трудоёмкости совместной работы в проекте различных категорий пользователей (инженеров-онтологов, которым требуется редактор с акцентом на профессиональные инструменты, экспертам предметных областей, которым требуется максимально простой и понятный визуальный редактор и разработчикам онтологически управляемых решений). С другой стороны, благодаря конфигурированию на уровне мета-онтологии, удается автоматизировать труд разработчиков онтологий и расширить функциональные возможностей визуального редактора онтологий.

Библиографический список

1. Oberle D. How ontologies benefit enterprise applications / Semantic Web Journal, 2009, vol. 5, no. 6, pp. 473-491.
2. OBDA systems // URL: http://obdasystems.com/obda (дата обращения 30.05.2021).
3. Jastrzab T., Postanogov I.S. Ontology reuse as a means for fast prototyping of new concepts / Communications in Computer and Information Science, 2017, vol. 716, pp. 273-287.
4. Protégé // URL: https://protege.stanford.edu/ (дата обращения 30.05.2021).
5. Chuprina S.I., Nasraoui O. Using Ontology-based Adaptable Scientific Visualization and Cognitive Graphics Tools to Transform Traditional Information Systems into Intelligent Systems / Scientific Visualization, 2016, vol. 8, i. 1, pp. 23-44.
6. Чуприна С.И., Зиненко Д. В. ОНТОЛИС: адаптируемый визуальный редактор онтологий // Вестник Пермского университета. 2013. Вып. 3. Математика. Механика. Информатика. С. 106-110.
7. Лапшин В.А. Онтологии в компьютерных системах. – 1-е изд. – М.: Научный мир, 2010. – 220 с.
8. Logux // URL: https://logux.io (дата обращения: 30.05.2021).
9. Marc Shapiro, Nuno Preguiça, Carlos Baquero, Marek Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. [Research Report] RR-7506, Inria – Centre Paris-Rocquencourt; INRIA. 2011, pp.50. ffinria-00555588
10. OWL 2 Web Ontology Language. Document Overview (Second Edition): World Wide Web Consortium (W3C) // URL: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/ (дата обращения 30.05.2021).

ONTSPACE AS AN EXAMPLE OF ADAPTABLE VISUAL ONTOLOGY EDITORS' DEVELOPMENT

Shartsev Grigorii K.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, me@shgk.me

Abstract. In this paper we propose an approach to the development of adaptable visual ontology editors by example of the development of web ontology editor "OntSpace". The concept is based on a model-driven method of design and development of intellectual systems and use a metaphor of Solar System components to concept representation. The metamodel of the ontology editor itself is the special kind of ontology – metaontology. This metamodel describes the knowledge about editor features, methods of visualization of vertices and edges of graph representation of ontology and consists of three layers: a core, profiles, and plugins. The core describes the editor's knowledge about itself and it's metamodel. Profiles represent knowledge for solving different kind of problems, and plugins represent knowledge for solving particular tasks. It allows to expand ontology editors features in a unified manner by means of editing the meta-ontology without modifying source code. Ontology driven approach allows to adapt editor to the preferences of different categories of users and takes into account the specific of subject area. Mechanism of ontology editor embedding to third party applications is presented.

Keywords: ontology editor, ontology; metamodel; meta-ontology; adaptable visual ontology editor, ontology-driven solutions.

УДК 004.891.3

РАЗРАБОТКА НЕИНВАЗИЙНОГО ИНТЕРФЕЙСА МОЗГ-КОМПЬЮТЕР ДЛЯ РАСПОЗНАНИЯ ЭМОЦИЙ НА ОСНОВЕ ЭЭГ

Юрков Михаил Александрович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, mishayur1997@mail.ru

В статье представлено описание разработки неинвазивного интерфейса мозг-компьютер для распознания эмоций на основе электроэнцефалограммы (ЭЭГ). Система позволяет распознавать 4 эмоции по данным ЭЭГ. Классификатор эмоций требует предобработку данных ЭЭГ, в тексте описаны используемые для этого методы, а также методы, используемые для обучения классификатора эмоций. Для тестирования использовались открытые данные с OpenNeuro.

Ключевые слова: машинное обучение, электроэнцефалограмма, распознание эмоций, неинвазивный интерфейс.

Эмоции играют большую роль во многих аспектах жизни человека, они могут влиять на процесс обучения и поведение в тех или иных ситуациях. Распознание эмоций может помочь эмоциональному осознанию личности человека [1], [2]. Оценка эмоций может помочь понять, как они влияют на различные действия или ощущения. Одним из примеров является эмоциональный маркетинг [3], где компании стараются стимулировать покупательскую способность не характеристиками товаров, а положительными отзывами,

бонусами от покупки товаров и профессионально подготовленных продавцов. Все эти аспекты создают положительные эмоции, что в свою очередь выделяет товар на фоне других.

На текущий момент наиболее распространено распознание эмоций по выражению лица. Однако у этого метода есть значительные недостатки: какие-либо тёмные очки или маски могут значительно ухудшить результат распознавания; человек хорошо контролирующий свою мимику может обмануть такие классификаторы.

Одним из набирающих популярность исследований является распознание эмоций на основе электроэнцефалографии (ЭЭГ). По сравнению с распознанием по выражению лица, этот метод сложно обмануть, так как считываются сигналы с мозга, которые у всех людей имеют схожую структуру.

В настоящее время наиболее распространено применение ЭЭГ для считывания моторных импульсов (воображаемые действия). При этом определение эмоций человека с помощью интерфейса мозг-компьютер в данный момент имеет меньшее количество реализаций [4]. Публикации на эту тему так же весьма немногочисленны [5].

Данные для классификации были взяты из источника OpenNeuro [6], набор данных называется «Imagined Emotion Study». Он создан в июле 2020 года. Данные записаны институтом нейронных вычислений в Сан-Диего. Запись была произведена для 34 субъектов.

Для формирования данных использовался электроэнцефалограф с 224 электродами, которые регистрируют данные с частотой дискретизации 512 Гц.

Условия проведения эксперимента – испытуемому включают звуковою инструкцию, каким образом будет производится эксперимент. В каждом эпизоде будет предложен один или несколько воображаемых сценариев. Сценарии будут описывать возможные обстоятельства, которые могут вызвать предлагаемые эмоции, и телесные ощущения, часто связанные с ними.

Испытуемый сидит в расслабленном состоянии и с закрытыми глазами. Как только испытуемый чувствует необходимую эмоцию, он должен нажать на ёмкостную кнопку пальцем. Данную процедуру можно повторить столько раз, сколько испытуемому захочется. Для переключения на инструкции к другой эмоции требуется нажать кнопку выхода. После каждого вида эмоции даётся пауза в минуту, чтобы вернуться в расслабленное состояние.

В структуре данных визуализации мозга (Brain Imaging Data Structure или BIDS) [7] описывается простой и легкий в использовании способ организации нейровизуализационных и поведенческих данных. Этот стандарт будет использоваться для облегчения воспроизводимости эксперимента.

Для чтения и предобработки электроэнцефалограмм используется библиотека MNE [8]. Эта библиотека имеет встроенные функции для чтения файлов электроэнцефалограмм в различных форматах, позволяет перенести вычисления на видеокарту, что в некоторых случаях может очень сильно ускорить процесс обработки данных и обучения классификаторов. Ещё одной важной особенностью MNE является встроенная поддержка работы с BIDS.

На рисунке 1 можно увидеть расположение электродов в данных, требуется выделить из этого набора электроды, совпадающие по месторасположению со схемой 10-20.

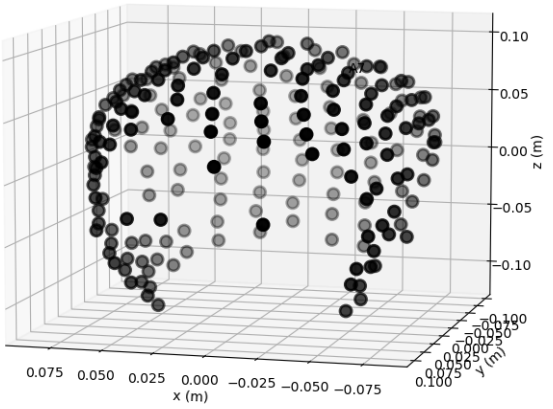


Рис. 1 – Расположение электродов в 3Д

Радиальная схема не позволяет воссоздать в точности схему 10-20, тем не менее мы воспользуемся тем, что у нас много электродов и будем использовать 36 из них. Электроды выбираются по принципу близости к расположению в схеме 10-20. На рисунке 2 представлено 2д и 3д отображение электродов.

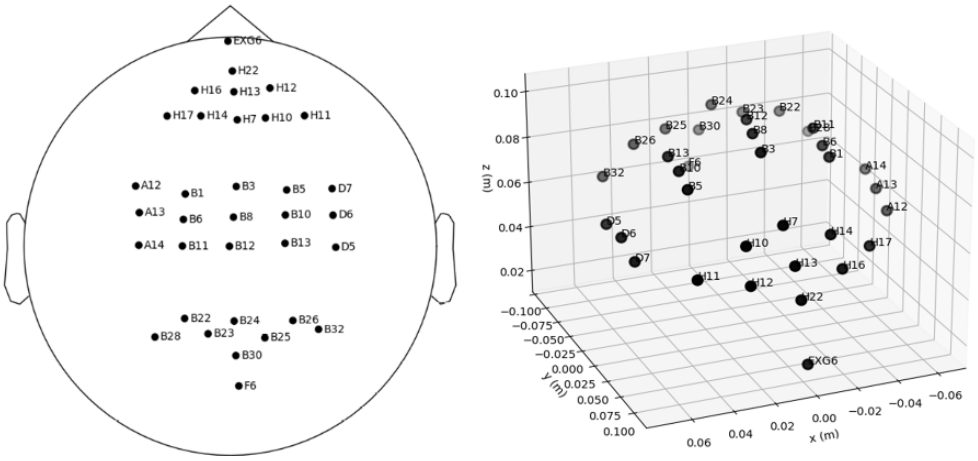


Рис. 2- Расположение выбранных электродов

Данные содержат аннотацию к временным рядам, т.е. в них помечено, в какой момент воспроизводился какой-либо стимул. Используя встроенные функции MNE мы создаём список событий на основе этих аннотаций.

Для избавления от мелких шумов применяется фильтр высоких частот с указанной частотой в 1 Гц, результат можно увидеть на рисунке 3.

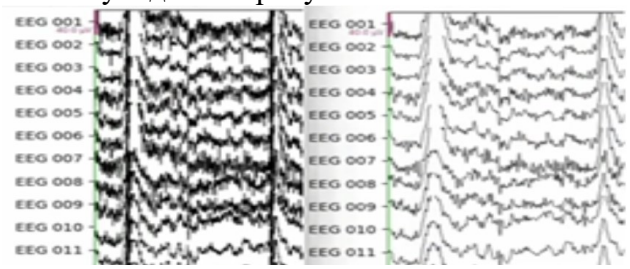


Рис. 3- Результат применения фильтра высоких частот

Для дальнейшего анализа данных нам потребуется выделить эпохи для каждого события. Под эпохой понимается часть данных с фиксированным временем до возникновения стимула и после него. Экспериментальным путём было выяснено, что начинать эпоху нужно за 0,7 секунды до возникновения стимула и заканчивать спустя секунду после стимула.

Далее убедимся, что различные эмоции имеют различные ЭЭГ. Для более яркого контраста сравним счастье и злость, для этого усредняем значения по всем эпохам злости и

счастья и строим два графика изменения электрического потенциала со временем. Результат можно увидеть на рисунке 4. Мы видим, что графики значительно отличаются, счастье имеет более высокие значение электрического потенциала.

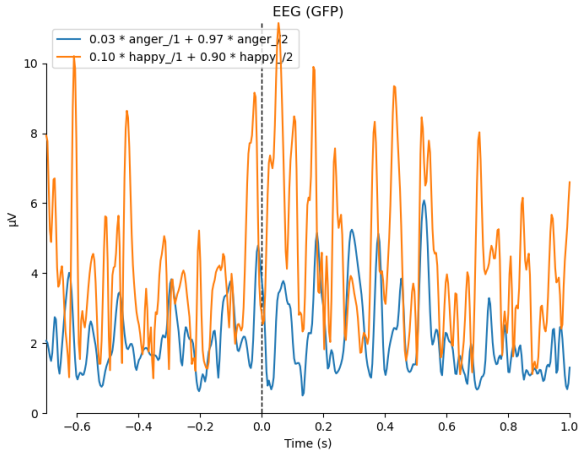


Рис. 4- Сравнение изменения напряжения в злости и счастья

Также мы можем посмотреть топологические карты электрического потенциала в определённые моменты времени. Снова будем рассматривать злость и счастье. На рисунке 5 можно увидеть топологические карты средней электрического потенциала. Нижняя полоса – злость, верхняя счастье.

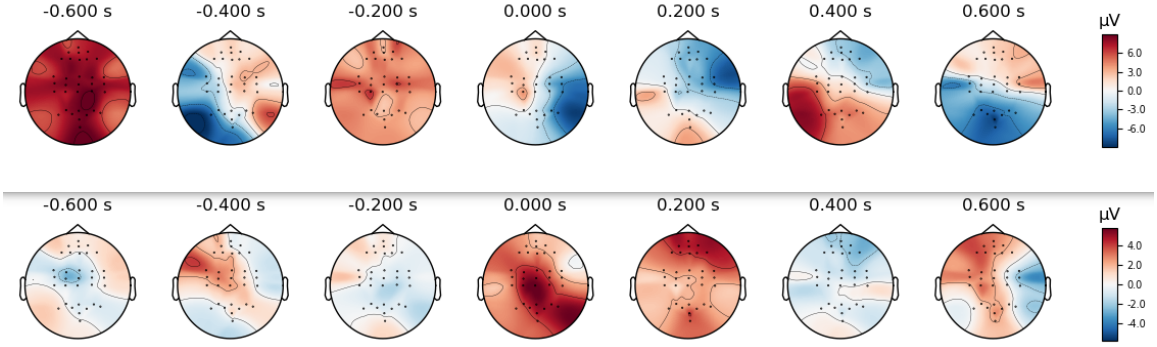


Рис. 5 – Топографически карты электрического потенциала по времени (сверху счастье, снизу злость)

В 0.2 секунды повышается правосторонняя лобная активность, что, как мы знаем из анализа данных, связано с негативными эмоциями. Для положительных эмоций должна быть левосторонняя лобная активность, но она выражена куда слабее по сравнению с негативными эмоциями.

После разделения данных на эпохи и некоторой фильтрации нужно постараться избавится от артефактов движения глаз, потому что даже с закрытыми веками глазные яблоки могут двигаться. Для этого пригодиться EOG канал, который ранее упоминался. Для наших данных передаём 0,9 в переменную количества основных компонентов. Это значит, что будет выбрано наименьшее кол-во компонентов, которые в совокупности смогут объяснить 90% компонентов. В качестве метода будем использовать «Пикард» [9], так как он имеет наиболее быструю сходимость.

Разделение эпох на диапазоны происходит с помощью встроенной в MNE функции построения спектральной карты мощности по каналам. Результат показан на примере эмоции злости.

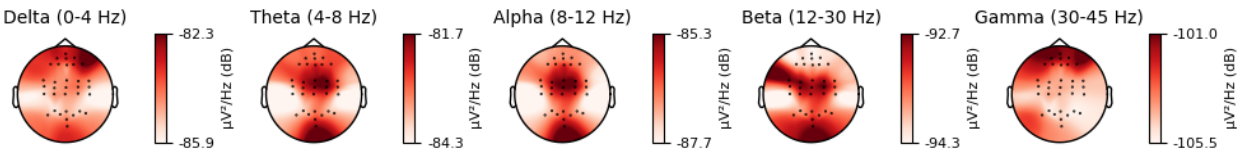


Рис. 6- Среднее значение диапазонов для злости

Классификатор планируется обучать на спектральных плотностях мощностей (power spectrum density или PSD), по каждому из ранее определённых диапазонов частот. Исходя из анализа литературы, это наиболее эффективный способ представления данных для классификации.

Для вычисления спектральной плотности мощности используется встроенная в MNE функция. Она использует метод Уэлча [10].

В качестве классификатора выбран метод опорных векторов (support vector machine или SVM). Как показал анализ литературы, SVM имеет наибольшую эффективность, это было подтверждено на практике. SVM метод выдал наиболее точные результаты. В качестве ядра была взята радиально базисная функция. Она превосходила в точности классификации остальные ядра в среднем на 10%.

В результате обучения классификатора была получена точность в 79,4%. В таблице ниже представлены метрики по распознаванию эмоций.

Таблица – Метрики распознавания эмоций

	Precision	Recall	f1-score	Всего
Гнев	0,75	0,82	0,78	11
Сострадание	0,78	0,70	0,74	10
Страх	0,85	0,85	0,85	13
Среднее арифметическое взвешенное	0,79	0,79	0,79	34

В результате реализации классификации были выявлены серьёзные недостатки в данных, с которыми ведётся работа. Поэтому следующей задачей будет сбор собственных данных для обработки. Это позволит контролировать количество записанных эмоций, а также использовать все нужные электроды при записи данных, что в свою очередь позволит более качественно балансировать классы, а также использовать обученный классификатор на данных любого испытуемого.

Библиографический список

1. H. Huang, Q. Xie, J. Pan, Y. He, Z. Wen, R. Yu, Y. Li An EEG-Based Brain Computer Interface for Emotion Recognition and Its Application in Patients with Disorder of Consciousness // IEEE Transactions on Affective Computing. – 2018.
2. A. Makhkamova, P. Ziegler and D. Werth Augmenting Collaboration with Invisible Data: Brain-Computer Interface for Emotional Awareness // in Mensch und Computer. 2019.
3. Агафонова, А.Н. и А.О. Чванов, 2017. Эмоциональный маркетинг как ключ к иррациональному поведению потребителей. Научный форум: Экономика и менеджмент // Общество с ограниченной ответственностью "Международный центр науки и образования" (Москва), С. 15-20.
4. Shepelev, E., M. Lazurenko, N. Kiroy, V. Aslanyan and R. Bakhtin, 2018. A Novel Neural Network Approach to Creating a Brain–Computer Interface Based on the EEG Patterns of Voluntary Muscle Movements. Neuroscience and Behavioral Physiology, 48: 1145-1157.
5. Giamberardino, P., D. Iacoviello, G. Placidi, M. Polsinelli and M. Spezialetti, 2018. A Brain Computer Interface by EEG Signals from Self-induced Emotions. European Congress, 27.
6. Imagined Emotion Study [Электронный ресурс] URL: https://openneuro.org/datasets/ds003004/versions/1.1.0 (дата обращения: 02.04.2021).
7. Brain Imaging Data Structure [Электронный ресурс] URL: https://bids.neuroimaging.io/ (дата обращения: 02.04.2021).
8. MNE [Электронный ресурс] URL: https://mne.tools/stable/index.html (дата обращения: 02.04.2021).
9. H. J. Ricardo, Chapter 2 – First-order differential equations // A Modern Introduction to Differential Equations (Third edition). 2021. p. 27-109.

10. Алюнов Д.Ю. О МЕТОДАХ ОЦЕНИВАНИЯ ПАРАМЕТРОВ СИГНАЛА // Современные проблемы науки и образования. 2014. № 6.

DEVELOPMENT OF A NON-INVASIVE BRAIN-COMPUTER INTERFACE FOR EMOTION RECOGNITION BASED ON EEG

Yurkov Michael A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, mishayur1997@mail.ru

The article describes the development of a non-invasive brain-computer interface for recognizing emotions based on an electroencephalogram (EEG). The system allows you to recognize 4 emotions based on EEG data. The emotion classifier requires preprocessing of EEG data, the text describes the methods used for this, as well as the methods used to train the emotion classifier. The work used open data from OpenNeuro.

УДК 519-6, 004.421

РАСЧЕТ ХАРАКТЕРИСТИК ПЕРКОЛЯЦИОННОЙ МОДЕЛИ *K*-МЕРОВ НА ПЛОСКОСТИ

Теплых Полина Дмитриевна, Бузмакова Мария Михайловна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, pteplich@gmail.com

В работе рассматривается континуальная модель перколяционной системы равномерно распределенных, непересекающихся линейных *k*-меров на плоскости. *k*-меры − отрезки фиксированной длины, которые не могут пересекаться. Начало и ориентация k-мера – случайные числа, сгенерированные с помощью алгоритма «Вихрь Мерсенна». При моделировании используются периодические граничные условия по двум направлениям. Для модели ранее разработаны эффективные алгоритмы равномерного распределения *k*-меров на плоскости, дискретизации континуальной системы *k*-меров на плоскости, распределения *k*-меров по кластерам, поиска перколяционного кластера, определения среднего размера кластера и мощности перколяционного кластера.

Ключевые слова: математическое и компьютерное моделирование, линейные k-меры, теория перколяции, мощность перколяционного кластера, средний размер кластера.

Введение

Проблема перколяции была центром фокуса исследований статистической механики в течение нескольких десятилетий [1-5]. Теория перколяции – очень простой, но общий, мощный и полезный инструмент для описания широкого класса явлений, которые принято называть критическими. Теория помогает моделировать задачи из разных областей наук: биологических (обогащение полезных ископаемых (перколяторы, где переход: частичное/полное смачивание), медицинских (распространение болезней среди населения, где переходом будет передача инфекции контактным путем/эпидемии), физических и др. Теория перколяции имеет точки соприкосновения с рядом новых и перспективных направлений науки, например, перколяционные процессы могут приводить к самоорганизации и образованию структур, объекты, которые образуются при перколяции, являются фракталами. Несмотря на то, что в теории перколяции получен ряд строгих

результатов, а в ее применении достигнуты значительные успехи, она находится еще в процессе становления, многое еще предстоит понять, доказать, применить. В связи с вышесказанным возникает необходимость в дальнейших исследованиях различных моделей перколяции, для получения новых, ранее не известных фундаментальных сведений.

В рамках работы была исследована континуальная модель перколяционной системы равномерно распределенных линейных *k*-меров на плоскости с периодическими граничными условиями. Было показано характерное поведение среднего размера кластера и мощности перколяционного кластера.

Описание задачи

Была предложена континуальная модель перколяционной системы равномерно распределенных линейных *k*-меров на плоскости с периодическими граничными условиями. Линейные *k*-меры – отрезки фиксированной длины. В модели *k*-меры не могут пересекаться и должны удовлетворять периодическим граничным условиям. Математическую постановку настоящей модели можно представить в виде

$$M = \langle L, k, N, R\{x_i, y_i, d_i\}, K\rangle, i = \overline{1, N},$$

где L – размер области,

k – фиксированная длина *k*-мера,

N – количество *k*-меров,

x_i – координата начала *k*-мера по оси *x*, y_i – координата начала *k*-мера по оси y, d_i – ориентация *k*-мера,

K – количество испытаний.

Было проведено компьютерное моделирование по определению значения порога перколяции при различных значениях *k*. Для каждого значения длины были определены пороги перколяции для систем конечных размерностей. Далее по полученным значениям порога перколяции для систем конечных размерностей были определены пороги перколяции для случая бесконечных систем. Кроме значений порога перколяции было получено поведение среднего размера кластера и мощности перколяционного кластера.

$S = \sum_s s\,\omega_s = \frac{\sum_s s^2 n_s(p)}{\sum_s s n_s(p)}$ – средний размер кластера (при вычислении среднего размера кластера не учитывается перколяционный кластер и суммирование проходит по всем кластерам, кроме перколяционного). Средний размер кластера S вблизи порога перколяции ведет себя как показательная функция: $S(p) \propto |p - p_c|^{-\gamma}$, где γ – универсальный критический показатель, не зависящий от вида решетки и типа перколяции, а зависящий только от размерности пространства задачи. Символ $\propto$ означает пропорциональность в пределе при $p \to p_c$.

$P_\infty(p) = \frac{N_\infty}{N}$ – мощность перколяционного кластера (вероятность того, что случайным образом выбранный узел принадлежит бесконечному кластеру). Мощность перколяционного кластера P_∞ вблизи порога перколяции ведет себя как показательная функция с критическим показателем β. $P_\infty(p) \propto |p - p_c|^{\beta}$, где β – универсальный показатель, не зависящий от вида решетки и типа перколяции, а зависящий только от размерности пространства задачи.

Результаты работы

Было проведено моделирование при входных параметра: L – разные размеры в зависимости от длины *k*-мера (для определения порога перколяции по 3-4 размерности для каждого значения длины *k*-мера); k = 10, 30, 50, 70, 100; K = 100. Была получены значения порога перколяции для всех рассматриваемых длин *k*-мера, которые представлены в таблице 1 и на рисунке 1.

Таблица 1. Значения порога перколяции при различных значениях длины *k*-мера

k	p_c
10	0.152 ± 0.004
30	0.147 ± 0.005
50	0.149 ± 0.007
70	0.156 ± 0.001
100	0.175 ± 0.009

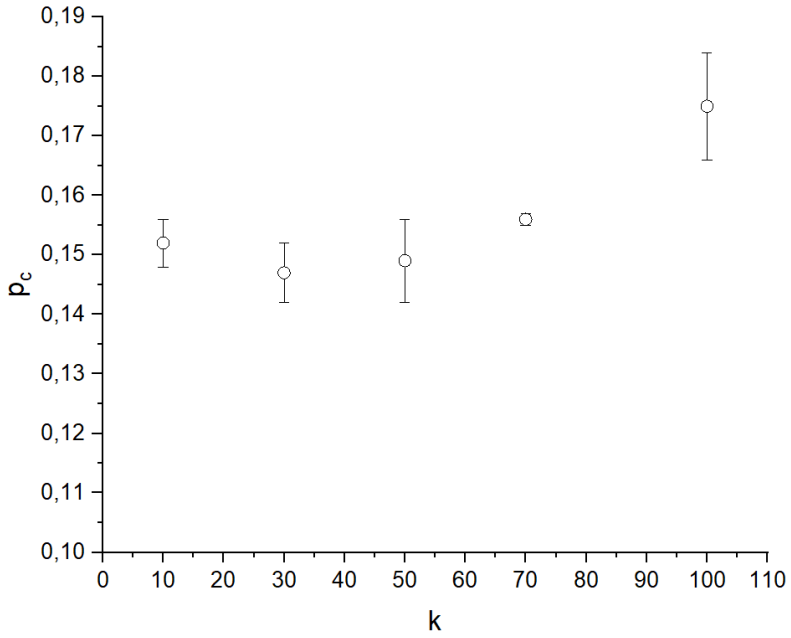


Рис.1. Значения порога перколяции при различных значениях длины *k*-мера

Из полученных значений можно утверждать, что поведение порога перколяции в континууме при увеличении длины *k*-мера аналогично поведению порога перколяции на решетках: что при увеличении длины *k*-мера до определенного значения порог перколяции снижается, а далее начинает снова увеличиваться. В перспективе определение значение длины *k*-мера, при которой порог перколяции перестает уменьшаться и начинает возрастать.

Также было получено характерное поведение среднего размера кластера и мощности перколяционного кластера (рисунки 2 и 3). Для данных характеристик были определены по стандартной в теории перколяции методике критические показатели (рисунки 4-6). Значения показателей для мощности перколяционного кластера после порога перколяции и среднего размера кластера до порога перколяции близки к известным значениям. Это подтверждает адекватность предложенной модели.

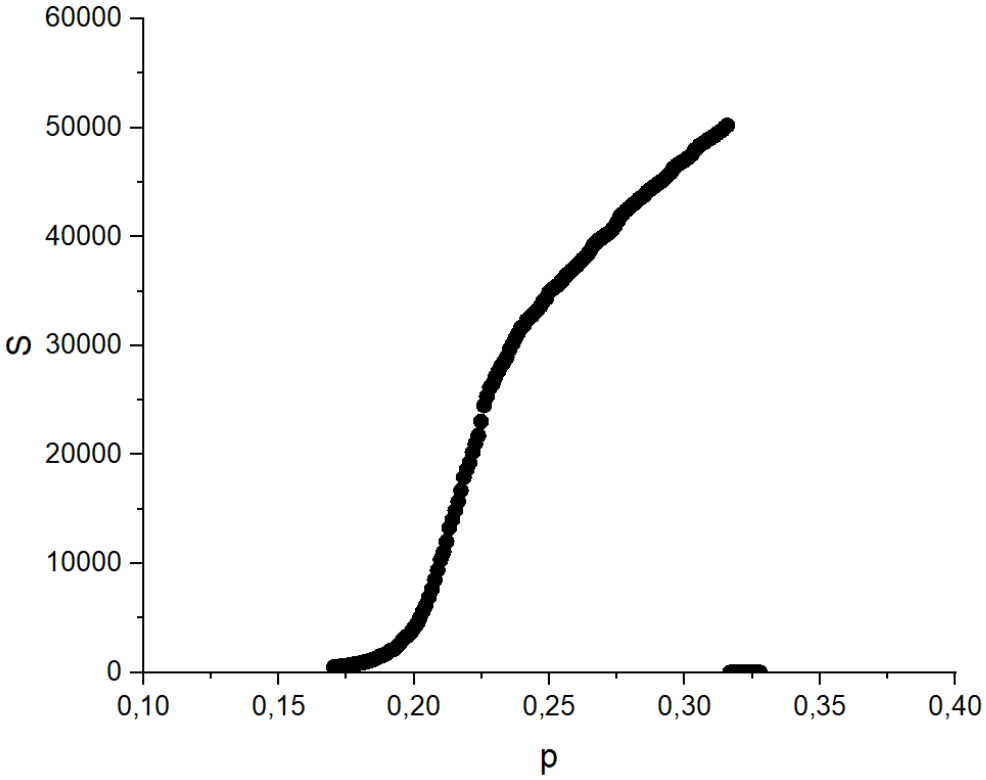


Рис. 2. Характерное поведение среднего размера кластера (*L* = 1000, *k* =50)

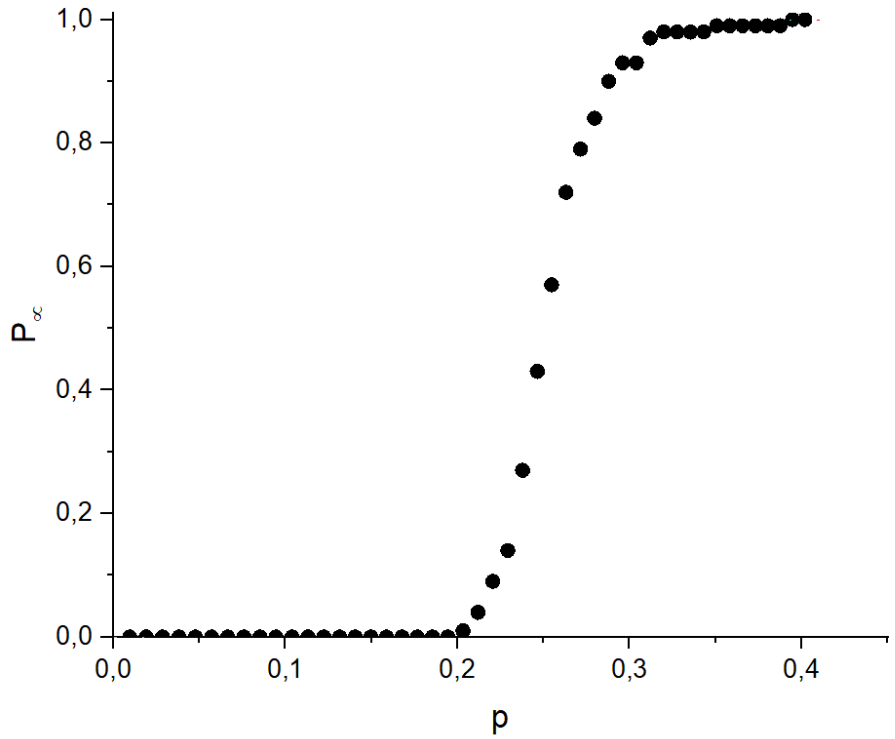


Рис. 3. Характерное поведение мощности перколяционного кластера (*L* = 1000, *k* =50)

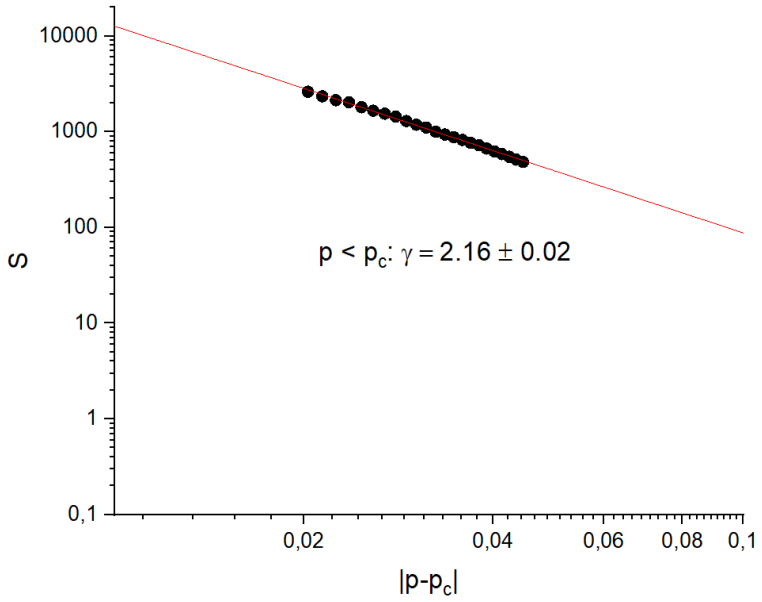


Рис. 4. Определение критического показателя для среднего размера кластера до порога перколяции (*L* = 1000, *k* =50)

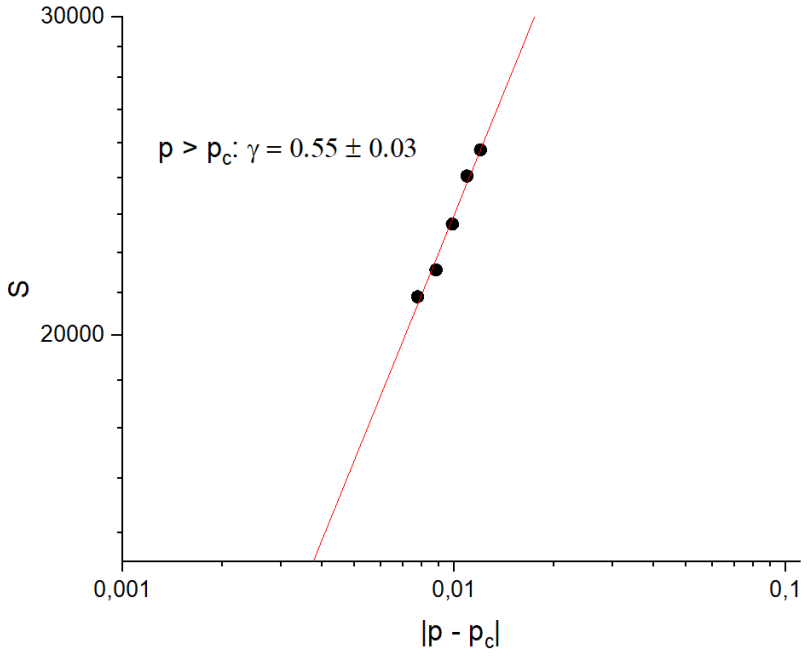


Рис. 5. Определение критического показателя для среднего размера кластера после порога перколяции (*L* = 1000, *k* =50)

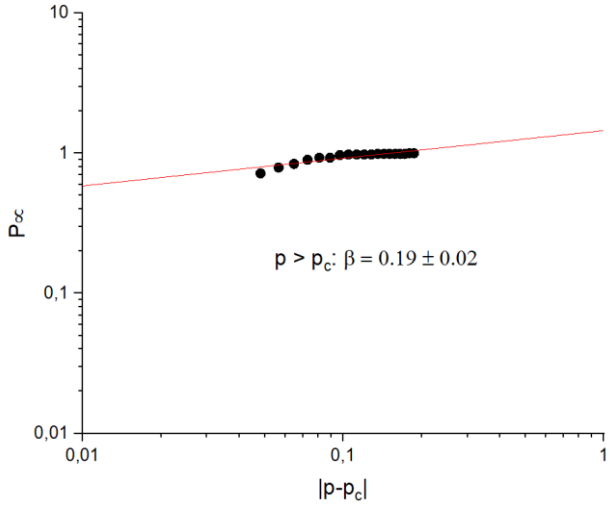


Рис. 6. Определение критического показателя для мощности перколяционного кластера после порога перколяции (*L* = 1000, *k* =50)

Выводы

Поведение порога перколяции в континууме при увеличении длины *k*-мера аналогично поведению порога перколяции на решетках: что при увеличении длины *k*-мера до определенного значения порог перколяции снижается, а далее начинает снова увеличиваться. Также получено характерное поведение среднего размера кластера и мощности перколяционного кластера при различных параметрах модели. Полученное поведение характеристик подтверждает корректность предложенной модели.

Библиографический список

1. *J.M. Hammersley,* Proc. Cambridge Phil. Soc. 1957. P. 53, 642.
2. *D. Stauffer,*Introduction to Percolation Theory // Taylor & Francis. 1985.
3. *R. Zallen,*The Physics of Amorphous Solids //John Willey & Sons, NY. 1983.
4. *J.W. Essam,* Reports on Progress in Physics. 1980. Vol. 43, 833.
5. *K. Binder,* Reports on Progress in Physics. 1997. Vol. 60, 488.
6. *C.Lorenz,R.May,R.Ziff,J.Stat.*Phys. 2000. V. 98, 961.
7. *Ю.Ю.Тарасевич* Перколяция: теория, приложения, алгоритмы М.:Едиториал УРСС, 2002. С. 12.

CONTINUOUS PERCOLATION OF K-MERS ON THE PLANE. CALCULATION OF PERCOLATION MODEL CHARACTERISTICS

Teplykh Polina D., Buzmakova Mariya M.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, pteplich@gmail.com

Abstarct. The continuum model of the percolation system consist of uniformly distributed linear *k*-mers is considered. In the model, *k*-mers are segments of fixed length, which cannot intersect. The origin and orientation of the *k*-mers are random numbers, generated using the Mersenne Twister' algorithm. The modeling uses periodic boundary conditions in two directions. For the model, efficient algorithms have been developed for the uniform *k*-mers distribution, for the discretization of continuum, for the *k*-mers distribution over clusters and for the search of pecolation threshold value. Also values of the average cluster size and the percolation cluster strength have been calculated.

Keywords: mathematical and computer modeling, linear *k*-mers, percolation theory, percolation cluster strength, average cluster size.

УДК 004.89; 004.4

РАЗРАБОТКА СИСТЕМЫ СБОРА И АГРЕГАЦИИ ПОКАЗАТЕЛЕЙ ЗДОРОВЬЯ ЧЕЛОВЕКА

Третьяков Александр Владимирович, Постаногов Игорь Сергеевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, tretyackov.aleks2012@yandex.ru

В данной статье представлена спроектированная и реализованная система сбора и агрегации показателей здоровья человека. Перечислены популярные аналогичные существующие системы, подробно описаны их недостатки. Перечислены и обоснованы средства реализации системы. Описана архитектура системы, вход/выход, а также кратко описан интерфейс взаимодействия с точки зрения пользователя и сторонней системы мониторинга здоровья. Изложена возможность добавления записей пульса, верхнего и нижнего давления по фотографии дисплея тонометра с произведенными измерениями. Описаны различные источники данных для системы. Проиллюстрирован пример отчетов, которые генерируются системой. Система может быть использована в других предметных областях, в которых необходимо выполнять следующие действия со множеством измерений каких-либо показателей: собирать, хранить, отслеживать динамику, выгружать собранные данные в файл, генерировать отчеты.

Ключевые слова: показатели здоровья человека, веб-приложение.

На данный момент приложения для здоровья довольно популярны. Основное предназначение этого класса приложений – сбор данных для построения картины состояния здоровья человека. В качестве самых известных можно назвать GoogleFit, Apple Health, Samsung Health. Однако в этих приложениях существуют недостатки.

Во время использования подобных приложений возможна ситуация, когда пользователь хочет отслеживать показатель здоровья, которого нет в приложении вообще. Для решения данной проблемы предлагается предоставить пользователю возможность добавлять в системе свои показатели здоровья. Однако на текущий момент времени добавлять новые записи для пользовательских показателей в системе можно только вручную.

Для того чтобы получить объективное описание состояния здоровья человека нужна консультация с врачом. Врачу необходимо видеть, как менялись показатели здоровья человека в течение времени, а не только на момент приема. Чтобы предоставить динамику своих показателей здоровья врачу можно показать графики в популярных приложениях, упомянутых ранее. Но это неудобно, так как графики различных показателей обычно расположены в разных разделах, а, может быть, и вообще в разных приложениях. Такой способ недостаточно нагляден и к тому же тратит время, которое в дефиците у врачей, на поиск нужных графиков. Предлагается демонстрация динамики показателей с помощью распечатанных отчетов, в которых будет отображена вся необходимая для врача статистика в наглядном виде.

Может возникнуть ситуация, когда пользователь захочет получить все данные, которые он хранит в приложении. Однако не во всех приложениях существует такая возможность. В разрабатываемой системе планируется сделать функцию выгрузки данных предельно доступной.

Было предложено разработать такую систему, в которой не будет недостатков, описанных выше. Для системы планируется сделать разные источники данных.

С точки зрения архитектуры, разрабатываемая система состоит из следующих частей: база данных; веб сервис с сайтом для взаимодействия с пользователями и API для импорта информации из сторонних систем; модуль для распознавания данных на фотографии дисплея тонометра.

С точки зрения входных/выходных данных, приложение имеет структуру, изображенную на рис. 1.

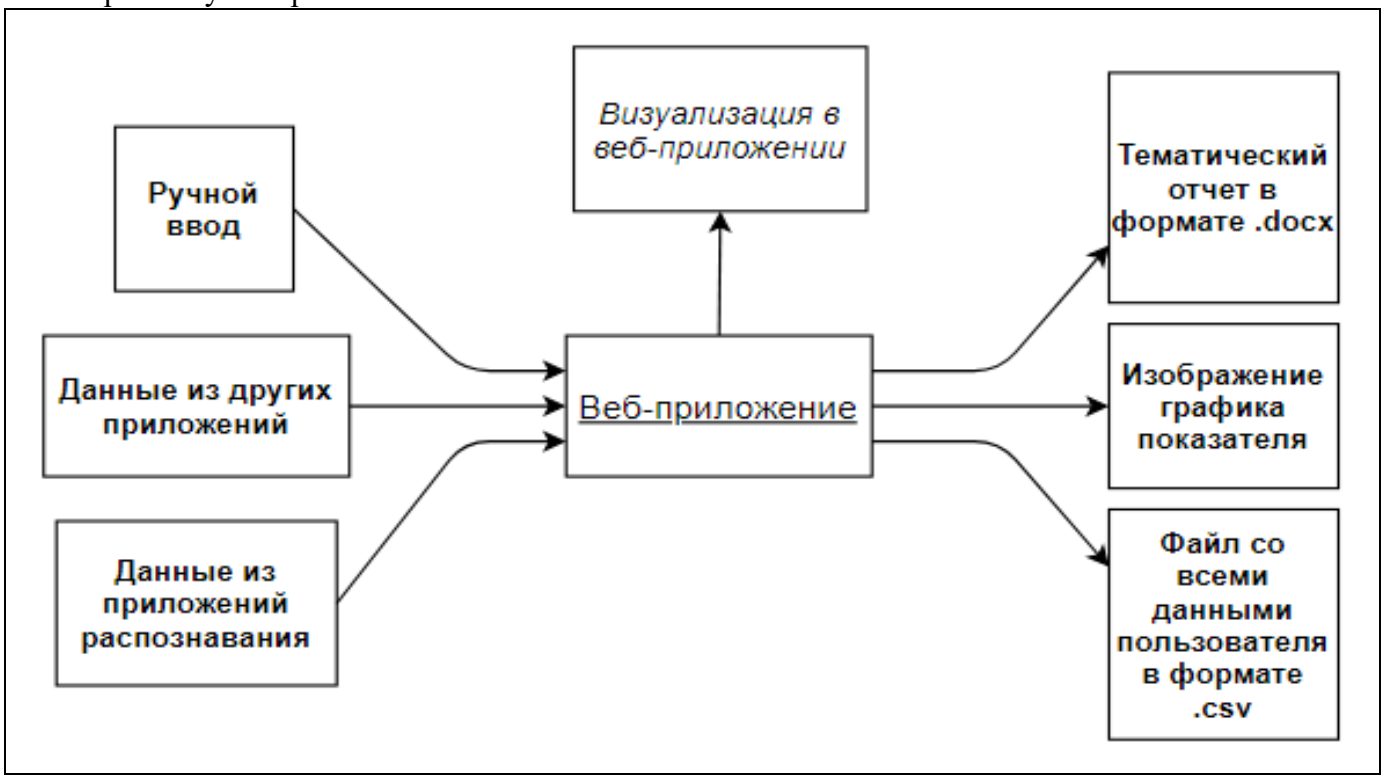


Рис. 1. Веб приложение с точки зрения входа/выхода

В качестве типа системы было выбрано веб приложение, так как в таком случае есть возможность использования разных типов устройств, например, смартфонов, планшетов, компьютеров. Для реализации веб приложения был выбран свободно-распространяемый кросс-платформенный фреймворк ASP.NET CORE, так как в таком случае не важно, какая ОС установлена на сервере. Для реализации модуля распознавания данных на фотографии дисплея тонометра был выбран язык программирования Python, так как для него существует больше количество готовых библиотек для распознавания.

Интерфейс для взаимодействия со сторонними системами предоставляет системам возможности записи измерений показателей в наше приложение и получения графика с динамикой показателя пользователя в виде изображения.

Интерфейс для взаимодействия с пользователями выполнен в виде сайта. Графический интерфейс главной страницы сайта продемонстрирован на рис. 2. На главной странице расположена краткая сводка показателей здоровья, которые отслеживаются у пользователя.

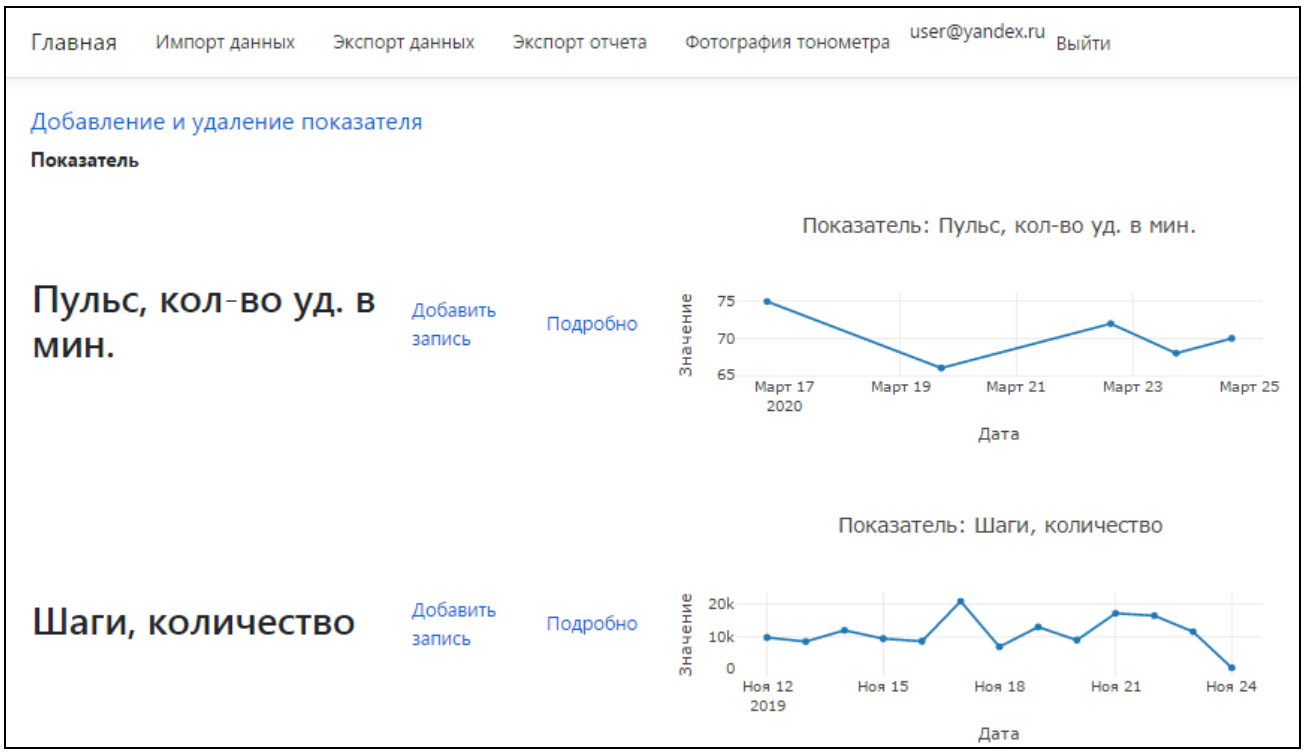


Рис. 2. Графический интерфейс главной страницы сайта веб приложения

Существует приложение «МТС 120/80» [1], которое считывает данные с фотографии дисплея тонометра. Эта функция является полезной, так как избавляет пользователя от ручного ввода. В нашем веб-приложении была реализована функция распознавания значений на дисплее тонометра. Был применен алгоритм распознавания 7-ми сегментных дисплеев [3]. В ходе алгоритма к фотографии применяется ряд фильтров из библиотеки OpenCV [2] и находятся контуры цифр, которые далее группируются в числовые значения конкретных показателей. В этом случае демонстрируется источник данных для веб-приложения в виде изображения.

В разрабатываемое приложение возможно загрузить данные по количеству шагов из резервной копии Google Fit. В этом случае файл формата csv используется как источник данных для системы.

Нами было доработано стороннее приложение шагомера для смартфонов, Pedometer[1], а именно добавлена возможность отправки собранных данных по сети в разрабатываемое веб приложение для демонстрации сбора измерений показателей с датчиков устройств.

Пример результата генерации тематического отчета изображен на рис. 3.

Рис. 3. Пример тематического отчета

В перспективе планируется добавить функционал, который при отклонении от нормы будет предупреждать человека о возможности заболевания и давать рекомендации, например, провести для уточнения диагноза измерения других показателей или вообще посетить врача.

Библиографический список

1. Пресс-релиз приложения «МТС 120/80» [Электронный ресурс] URL: https://perm.mts.ru/about/media-centr/soobshheniya-kompanii/novosti-mts-v-rossii-i-mire/2020-01-29/mts-sozdala-besplatnoe-prilozhenie-mts-12080-dlya-zaboty-o-serdce (дата обращения: 2.02.2020).
2. OpenCV [Электронный ресурс] URL: https://opencv.org/ (дата обращения: 3.02.2020).
3. Recognizing digits with OpenCV and Python [Электронный ресурс] URL: https://www.pyimagesearch.com/2017/02/13/recognizing-digits-with-opencv-and-python (дата обращения: 3.02.2020).
4. Pedometer [Электронный ресурс] URL: https://github.com/j4velin/Pedometer (дата обращения: 6.01.2020).

DEVELOPMENT OF A SYSTEM FOR COLLECTING AND AGGREGATING PARAMETERS OF HUMAN HEALTH

Tretyackov Alexander V., Postanogov Igor S.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, tretyackov.aleks2012@yandex.ru

We present a designed and implemented system for collecting and aggregating parameters of human health. Popular analogous existing systems are listed, and their disadvantages are described in detail. The means of implementing the system are listed and justified. The system architecture is described. System inputs and outputs are displayed. The interaction interface is briefly described from the point of view of the user and the external health monitoring systems. The possibility of adding pulse, upper pressure and lower pressure records based on the photo of the tonometer display with the measurements made is outlined. Different data sources for the system are described. An example of reports generated by the system is illustrated. The system can be used in other subject areas where you need to perform the following actions with multiple measurements of any indicators: collect, store, track dynamics, upload the collected data to a file, and generate reports.

УДК 004.413.2

АНАЛИЗ АРХИТЕКТУР ФРЕЙМВОРКОВ ДЛЯ МОБИЛЬНОЙ РАЗРАБОТКИ

Яковлев Петр Анатольевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, yakovlev123peter@gmail.com

Целью данной работы является анализ Фреймворков мобильной разработки, а также мобильных операционных систем, в которых они используются. Существует подробный анализ ключевых компонентов архитектуры и API, используемых в Android и iOS, которые важны для передачи информации и данных между уровнями абстракции в мобильной ОС. В этой статье рассматриваются как нативные методы разработки, так и кроссплатформенные решения, и то, как они структурированы и построены. Рассматриваются ключевые библиотеки и API встроенные в ОС в качестве методов коммуникации мобильных приложений с модулями аппаратного обеспечения. Этот документ представляет собой общий взгляд на структуру и архитектуру мобильных Фреймворков, которые позже будут использованы в качестве основы для дальнейшей работы над анализом производительности каждых из выбранных Фреймворков.

Ключевые слова: архитектура, api (application programming interface), фреймворк, библиотека, нативная разработка, кросс-платформ.

Введение

За последние 15 лет наблюдается растущая тенденция доминирования мобильных приложений на рынке взаимодействия с пользователем, удобства и монетизации. Трудно найти бизнес или компанию, которые не используют в той или иной форме мобильные приложения. Помня об этом, для процветающего бизнеса важно выбирать, как разрабатывать и поддерживать свои мобильные приложения. Независимо от того, будет ли процесс разработки полностью передан на аутсорсинг или будет осуществляться собственными силами, выбор структуры для использования в процессе создания приложения будет иметь огромное и длительное влияние на успех приложения и, следовательно, на компанию, которая его создала.

Чтобы распространить среди большинства пользователей, мобильное приложение должно адаптироваться к двум платформам – Android и iOS. Различия между этими двумя платформами велики и часто требуют разных наборов навыков для разработки, таких как Java/Kotlin для Android и ObjC/Swift для iOS. Таким образом, разработчики и компании часто борются со сложностью разработки кроссплатформенных приложений.

Разработка отдельных приложений обойдется дороже, чем разработка кроссплатформенных приложений. Поскольку нативный метод разработки требует индивидуальных кодов для разных платформ, в то время как для кроссплатформенной среды нужен только один код для обеих платформ. Поскольку доступно множество вариантов, выбрать подходящую среду для разработки вашего приложения сложно. Общий цикл разработки зависит от выбора правильной структуры, правильной платформы, правильной методологии разработки и правильного анализа требований проекта [1].

Данная работа сосредоточена на анализ архитектур операционных систем, фреймворков разработки и их взаимодействие с друг другом. Эта информация дальше будет использоваться для разработки системы рекомендаций фреймворков для мобильной разработки.

Архитектура мобильных операционных систем

Операционные системы взаимодействуют с оборудованием устройства, обрабатывая все, от ввода с клавиатуры до Wi-Fi, устройств хранения и дисплеев. Затем связь с этими компонентами компьютера может быть установлена программным обеспечением более высокого уровня через программное обеспечение в операционной системе, называемое интерфейсом прикладного программирования (API). API – это способ предоставить компоненты системы другому программному обеспечению без того, чтобы человек, пишущий программное обеспечение, обязательно знал все детали того, как функционирует компонент.

Фреймворки могут взаимодействовать с API, предоставляемыми операционной системой, независимо от языка и знания внутреннего функционирования системы в целом. Таким образом, мы можем писать программное обеспечение, совместимое с несколькими операционными системами, независимо от того, Android это или iOS, потому что API, предоставляемые этими операционными системами, не зависят от языка, если вы правильно с ними взаимодействуете.

Архитектура iOS

Core OS Layer

- Самый нижний уровень архитектуры iOS это Core OS
- Обеспечивает обработку, сопряжение и поддержка безопасности для приложения iOS
- Содержит Фреймворки Accelerate framework, System framework, External Accessory and security framework

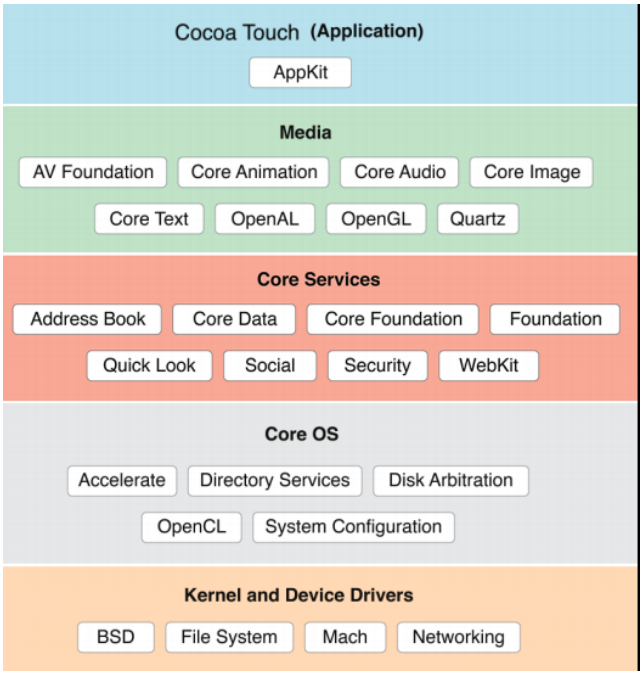


Рис 1 Архитектура iOS операционной системы

Cocoa Touch Layer

- Верхний слой iOS архитектуры
- Содержит ключевой Фреймворк UI kit
- Определяет базовую инфраструктуру приложений
- Обеспечивает основные функции, такие как многозадачность и сенсорный ввод
- Он содержит ключевые Фреймворки, такие как EventKit, GameKit, MapKit, MessageKit.

Core Service Layer

- Ниже слоя мультимедиа
- Обеспечивает услуги, используемые на верхних уровнях.
- Содержит Фреймворки Address Book framework, Core Location framework, Core Telephony framework

Media Service Layer

• Уровень мультимедиа предоставляет iOS все функции для поддержки аудио, видео, анимация и графика

• Он содержит Фреймворки Core Аудио, Core Text и Core Image.

Эта архитектура делит дизайн на мелкие части. Каждый слой добавляет услуги, предоставляемые нижними уровнями таким образом, что самый высокий уровень предоставляет полный набор услуг для управления коммуникациями и запуска приложения [3].

Приложения iOS не связываются напрямую с базовым оборудованием. Обсуждение приложений с оборудованием через набор четко определенных системных интерфейсов.

Архитектура Android OS

- Application framework – Фреймворк использованный для разработки приложений
- Binder IPC (Binder Inter-Process Communication) – Связь между процессами Binder позволяет платформе приложения пересекать границы процессов и вызывать код системных служб Android
- System services – Функциональность, предоставляемая API фреймворка приложения, взаимодействует с системными службами для доступа к базовому оборудованию
- Hardware abstraction layer (Уровень аппаратной абстракции) – Использование HAL позволяет реализовать функциональность, не затрагивая и не изменяя систему более высокого уровня.
- Linux kernel – Android использует версию ядра Linux с некоторыми специальными дополнениями

Рис 2 Архитектура Android операционной системы

Application Framework предоставляет приложениям множество высокоуровневых сервисов в форме классов Java [3].

- Activity Manager – контролирует все аспекты жизненного цикла приложения и стека действий
- Content Providers – позволяет приложениям публиковать данные и обмениваться ими с другими приложениями
- Resource Manager – предоставляет доступ к встроенным ресурсам без кода, таким как строки, настройки цвета и макеты пользовательского интерфейса
- Notifications Manager – позволяет приложениям отображать предупреждения и уведомления для пользователя
- View System – расширяемый набор представлений, используемых для создания пользовательских интерфейсов приложений

Так как в iOS закрытый исходный код то не все компоненты известны, но в android ключевой компонент который необходимо исследовать это называемый виртуальной машиной Dalvik или ART(Android Runtime), который представляет собой разновидность виртуальной машины Java, специально разработанную и оптимизированную для Android. В виртуальной машине Dalvik используются основные функции Linux, такие как управление памятью и многопоточность, присущие языку Java. Виртуальная машина Dalvik позволяет каждому приложению Android работать в собственном процессе с собственным экземпляром виртуальной машины Dalvik

Среда выполнения Android также предоставляет набор основных библиотек, которые позволяют разработчикам приложений Android создавать приложения Android с использованием стандартного языка программирования Java.

Многие основные компоненты и службы системы Android, такие как ART и HAL, созданы из собственного кода, для которого требуются собственные библиотеки,

написанные на C и C ++. Платформа Android предоставляет API-интерфейсы Java framework, чтобы предоставить приложениям функциональность некоторых из этих собственных библиотек. Например, вы можете получить доступ к OpenGL ES через API Java OpenGL платформы Android, чтобы добавить поддержку рисования и управления 2D- и 3D-графикой в вашем приложении.

Инструменты для нативной разработки мобильных приложений

Приложение общается с платформой для создания виджетов или доступа к таким службам, как камера. Виджеты отображаются на холсте экрана, а события передаются обратно в виджеты. Это простая архитектура, но приходится создавать отдельные приложения для каждой платформы, потому что виджеты разные, не говоря уже о родных языках.

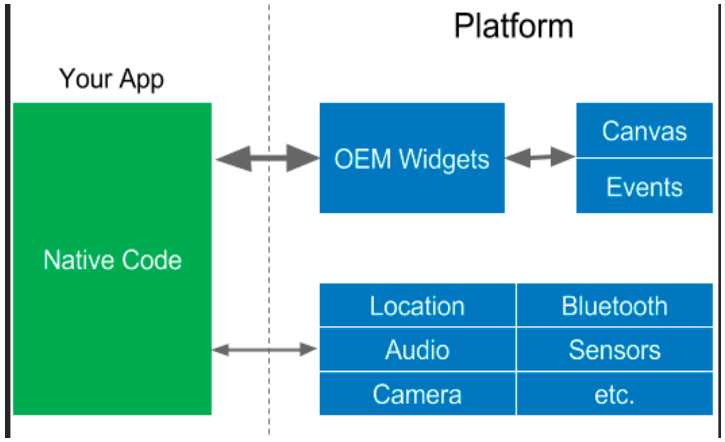


Рис 3 Взаимодействие нативных приложений с операционной системой

Архитектуры кросс-платформ Фреймворков для мобильной разработки

Существуют три основных вида кросс-платформ Фреймворков[4] –

1)Hybrid web application

- Гибридное веб-приложение – Подход использует движок браузера на устройстве и синхронизирует содержимое HTML в собственных веб-контейнерах, таких как WebView в android, UIWebView в iOS
- Эти веб-контейнеры имеют доступ к специфическим для платформы функциям через API
- Отсутствие нативных компонентов
- Медленная загрузка веб-контейнеров

2)Interpreted application

- Интерпретируемое приложение – может использовать популярные языки для разработки которые интерпретируются в нативный код платформы
- Используя интерпретацию, эти приложения могут иметь доступ к API платформы и тем самым использовать нативные компоненты
- Возникают проблемы с производительностью из-за преобразования кода во время выполнения

3)Cross-compiled application

- Код компилируется на платформа-специфичный, высокопроизводительный нативный код с помощью кросс-компилятора
- Могут обеспечить естественные ощущения и производительность, потому что у них есть доступ ко всем нативным элементам пользовательского интерфейса и API
- Специфические настройки платформы и конфигурации могут быть необходимы для доступа к возможностям устройства, таким как звук устройства и видео, календарные приложения, информация об устройстве и т. д.
- В этот раздел включаются React Native и Flutter

React Native

Архитектура приложения React Native состоит из виртуальной машины JavaScript, мост React Native и собственные модули. Код приложения запускается на виртуальной машине JS вместе с любыми используются сторонние библиотеки. Вызовы

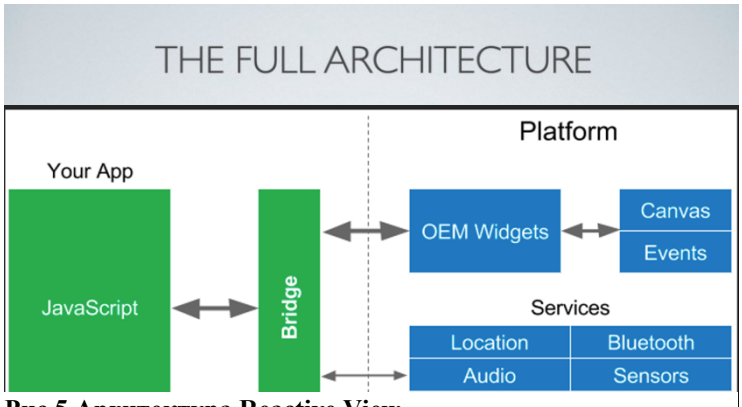


Рис 5 Архитектура Reactive View

собственных модулей направляются через React Native мост к собственным API и сторонним библиотекам, и результаты передаются если нужно, вернитесь через мост. Это позволяет использовать JavaScript для разработки приложение, по-прежнему используя собственные компоненты пользовательского интерфейса и функции платформы. Архитектура приложения основана на Facebook React, компонентной платформе для создания интерфейсов для веб-приложений. Базовый элемент Приложение React – это компонент, который представляет собой отдельный класс JavaScript, который всегда отображает часть пользовательского интерфейса. Компонент также может получать свойства и имеют внутреннее состояние. Несколько компонентов затем составлен для создания более сложных пользовательских интерфейсов, и данные могут быть переданы вниз цепочки компонентов в качестве свойств, от родителя к потомку. Из-за моста к собственным api приложения могут быть замедленны в производительности и времени начала приложения.

Flutter

Flutter имеет новую архитектуру, которая включает виджеты, которые хорошо выглядят и удобны, быстрые, настраиваемые и расширяемые. Flutter не использует виджеты платформы (или DOM WebViews), он предоставляет свои собственные виджеты. Виджеты без учета состояния или состояния являются строительными блоками любого приложения Flutter и могут быть тематически похожими на собственные компоненты пользовательского интерфейса Android (материал) или iOS (купертино). Виджеты отображаются на холсте Skia с поддержкой улучшенной анимации и распознавания жестов. Движок Flutter содержит основные технологии Skia – библиотеку рендеринга 2D-графики – и виртуальную машину на языке Dart в специальной платформе. Любая оболочка реализует соответствующие API-интерфейсы платформы и обрабатывает события жизненного цикла приложения системы[5]. Использование языка Dart позволяет Flutter заблаговременно компилировать исходный код в собственный код. Код C / C ++ движка скомпилирован с помощью Android NDK или iOS LLVM. Обе части обернуты в «бегущий» проект Android и iOS, что приводит к созданию файла apk или ipa соответственно. При запуске приложения любой рендеринг, ввод или событие делегируются скомпилированному движку Flutter и коду приложения. Необходимость упаковать движок в файл apk / ipa приложения в настоящее время приводит к увеличению размера приложения до 4 МБ. Быстрый запуск и выполнение приложения – это преимущества компиляции в нативный код. Пользовательский интерфейс обновляется со скоростью 60 кадров в секунду – в основном с использованием графического процессора – и каждый пиксель на экране принадлежит холсту Skia, что обеспечивает плавный, настраиваемый пользовательский интерфейс.

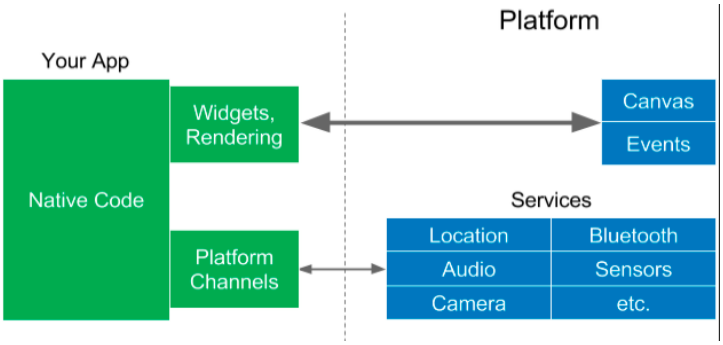


Рис 6 Архитектура Flutter

Заключение

Данная статья предоставляет детальный обзор архитектур и методов взаимодействий фреймворков для мобильных приложений с операционной системой на которой они работают. Используя данный обзор, будет выполнена дальнейшая работа по созданию приложений для тестирования производительности мобильных фреймворков и определены критерии их выбора.

Информация, предоставленная в этой статье, дает понимание о внутренней работе каждой структуры, а также понимание работы мобильных операционных систем, а также их структуру и расположение в контексте передачи информации и данных между уровнями абстракции.

Библиографический список

1. K. Bircan, Cross-Platform Mobile App Development with Flutter – Xamarin – React Native: A Performance Focused Comparison, Nov. 2017, Available (accessed 18.11.2018): https://medium.com/@korhanbircan/crossplatform – mobile – app – development – with – flutter – xamarin – react – native-a-performance-focused-a4457bcbdacc.
2. Apple, iOS Developer Documentation, Available: https: //developer.apple.com/.
3. Google, Android Developer Documentation, Available: https://developer.android.com/index.html.
4. C. P. R. Raj, S. B. Tolety, A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach, 2012 Annual IEEE India Conference (INDICON), ID: 1, 2012, pp. 625–629.
5. S. Ladd, Announcing Flutter beta 1: Build beautiful native apps, Feb. 2018, Available: https : / / medium . com / flutter – io / announcing-flutter-beta-1-build-beautiful-native-apps-dc142aea74c0.
6. Charkaoui S, Adraoui Z, Benlahmar EH. Cross-platform mobile development approaches. In: 2014 Third IEEE International Colloquium in Information Science and Technology (CIST) [Internet]. Tetouan, Morocco: IEEE; 2014 [cited 2020 Mar 16]. p. 188–91. Available from: http://ieeexplore.ieee.org/document/7016616/

ANALYSIS OF MOBILE DEVELOPMENT FRAMEWORK ARCHITECTURES

Yakovlev Petr A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia,yakovlev123peter@gmail.com

The purpose of this work is the analysis of mobile development frameworks as well as the mobile operating systems they are used in. There is a detailed analysis in key architecure components and API's used in Android and iOS which are vital for the communcation of information and data between levels of abstraction in the mobile OS. Native development as well as cross-platform solutions are looked at in this paper and the way in which they are structured and built is used in further work on my dissertation as a basis of which aspects of a mobile framework it is important to examine and test. This paper is an overall view into the structure and architecure of mobile frameworks which will be later used as a basis for further work on key insights into mobile development.

Keywords: architecture, api (application programming interface), Framework, library, native development, cross-platform.

УДК 004.9

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ОНЛАЙН-КОНСУЛЬТИРОВАНИЯ

Главатских Ксения Елизаровна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, glavatskikh.kseni@gmal.com

Аннотация. Данная статья посвящена проектированию и документированию информационной системы для онлайн-консультирования. Определена актуальность проектируемой системы. Проведён анализ подобных информационных систем, вследствие чего выделены достоинства и недостатки каждой. В результате анализа определены требования к информационной системе, на основании которых она спроектирована. Представлены: краткое описание предметной области, обзор возможных средств проектирования и моделирования информационных систем. На этапе проектирования построены модели разрабатываемой системы. Для этого используется программный инструментарий, объединенный под общим названием CASE-средства. Для разработки рассмотрены популярные средства реализации, а для хранения и работы с данными – СУБД, вследствие этого выбраны подходящие для проектируемой информационной системы. Созданы: модель поведения информационной системы, структурная и физическая модели, и представлен макет пользовательского интерфейса.

Ключевые слова: информационная система, онлайн-консультирование, чат, чат-бот.

Введение

Трудно поспорить с тем, что в последнее время интернет занимает немаловажную роль в жизни каждого человека, так как он значительно упростил коммуникацию между людьми, что, в свою очередь, способствует всё большему распространению использования интернета, например, как одного из инструментов для ведения бизнеса. Все больше компаний использует возможности виртуальных помощников, а именно виджетов чата и чат-ботов, для самых различных целей. Чат онлайн-консультирования – это простое и эффективное решение, дающее возможность оперативно, почти мгновенно решить проблему посетителя в текстовом виде, а главное – можно четко и правильно сформулировать вопрос и получить на него не менее четкий ответ.

Установленный на сайте онлайн-консультант способен повысить лояльность клиента к интернет-магазину, оказать влияние на решение покупателя и повысить конверсию сайта. Данный инструмент общения является дополнительным каналом взаимодействия интернет-магазина с клиентами [1, с.136].

Значимость создания чата онлайн-консультирования заключается в том, что пользователь получит гибкую настройку интерфейса и основных функций, требующихся при использовании данного чата. Это востребовано, так как многие компании порой пишут собственные чаты для своих нужд только из-за того, что их не устраивает графический интерфейс, представленный уже на рынке продуктами [2, с.4].

Краткий обзор средств проектирования

На этапе проектирования строятся модели разрабатываемой системы. Для этого используется программный инструментарий, объединенный под общим названием CASE-средства.

При выборе средств проектирования важную роль играла модель распространения, то есть – наличие постоянной бесплатной версии. По этому критерию выделены два средства – StarUML и Software Ideas Modeler. Обе программы имеют интуитивно понятный интерфейс, возможность бесплатного использования в некоммерческих целях, своевременное обновление и поддержку разработчиком.

Отметим, что необходимо также определить средство моделирования базы данных, в качестве инструмента которой используют ER-диаграмму, которая позволяет рассмотреть систему целиком и выяснить требования, необходимые для ее разработки, касающиеся хранения информации. Такая возможность есть в Software Ideas Modeler, на котором и сделан выбор в качестве средства проектирования. При моделировании базы данных ИС для онлайн-консультирования использована нотация Crow's Foot.

Краткий обзор технологий для разработки

Информационную систему онлайн-консультирования представим в виде веб-сервера, состоящего из БД, API, агента, управляемого через сайт, который будет являться интерфейсом системы и окна чата, которое будет размещаться на стороннем сайте. Для написания API был выбран язык программирования для web – PHP. Он хорошо документирован, имеет встроенные функции для работы с HTTP, прост в освоении и отладке. Также он имеет высокую степень переносимости и хорошо работает под управлением различных ОС.

Интерфейс нашей системы будет разработан при помощи языка разметки HTML. Связь интерфейса с API будет осуществляться через скрипты, написанные на JavaScript. Также потребуется связь баз данных и SQL. Всё это полноценно поддерживается PhpStorm.

Для оперативной связи между посетителем стороннего сайта и менеджером будем использовать WebSocket – протокол полнодуплексной связи (может передавать и принимать одновременно) поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени, и SignalR – библиотеку от компании Microsoft, предназначенную для работы с коммуникациями реального времени (SignalR использует в своей основе протокол WebSocket) [3].

Как упоминалось выше, необходимо работать с базами данных – создавать и управлять ими. Учитывая опыт работы с PostgreSQL и выделяя ключевой особенностью то, что необходимо будет работать с большими базами данных, остановимся в выборе на СУБД PostgreSQL.

Моделирование поведения ИС

В системе для онлайн консультирования будет 3 типа пользователей: менеджер, оператор и посетитель стороннего сайта. Последнему предоставляется только общение в чате. Первому доступно также общение в чате, только не со стороннего приложения или окна чата на посещаемом сайте, а с ИС. Помимо этого, он может изменять информацию о посетителях и управлять чатом. Менеджер имеет самый обширный доступ к ИС, ему предоставляются все возможности оператора, к которым добавляются управление операторами, новостями и настройками.

Для более подробного описания отношений между актерами и прецедентами разработана модель проектируемой системы в виде диаграммы вариантов использования (см. Прил. А.1). Также изображена диаграмма деятельности (см. Прил. А.2), поясняющая прецедент "Общение в чате". На ней показаны взаимодействия между посетителем, чат-ботом и оператором.

Моделирование структуры ИС

На данном этапе необходимо построить логическую модель ИС, которая будет описывать основные понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Для этого воспользуемся диаграммой сущность-связь (ERD).

Перед построением диаграммы опишем сущности, определяющие проектируемую систему. Сущность "Чат" включает в себя информацию о тех, кто взаимодействует в

общении. Сущность “Сообщение” содержит информацию о самом сообщении. Сущность “Пользователь” используется для идентификации всех взаимодействующих с системой лиц. Сущность “Консультант” содержит информацию обо всех операторах и менеджерах, имеющих доступ к ИС. Сущность “Посетитель” хранит информацию о посетителях стороннего сайта.

Построим диаграмму сущность-связь (ERD) в Software Ideas Modeler – Приложение Б.

Проектирование макета пользовательского интерфейса

Для создания макета интерфейса был использован онлайн-сервис “proto.io”, использующий UI перетаскивания, поэтому здесь не требуется кодировать. Определены следующие страницы личного кабинета менеджера и оператора: «Авторизация», «Чаты», «Операторы», «Новости», «Настройки», «Выход». Ниже представлена вкладка «Чаты» (см. рис. 1), где можно увидеть посетителей сайта, на котором установлен чат для онлайн-консультирования, чат с выбранным посетителем и информацию о нём. Также можно перейти в разделы “Мои чаты” и “Рабочие чаты”.

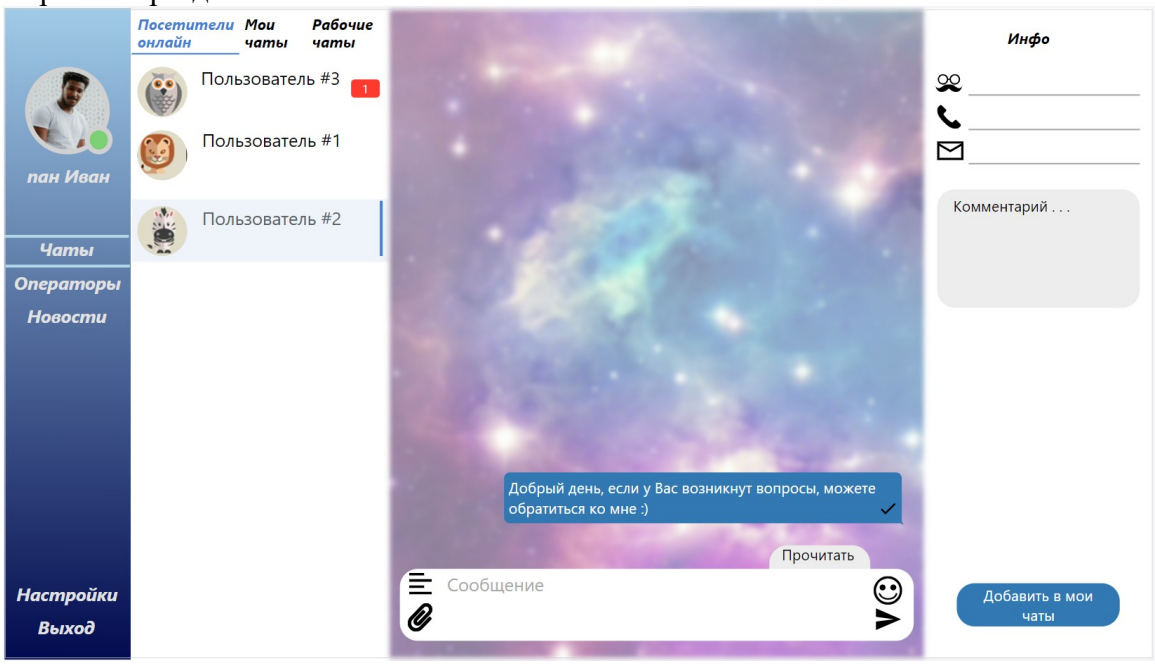


Рис. 1 – вкладка "Чаты"

Для большей наглядности внешнего вида окна чата и его возможностей, ниже представлены прототипы (см. рис. 2 и рис.3).

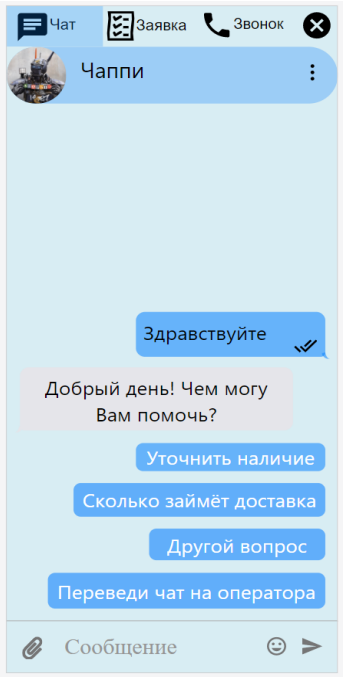


Рис. 2 – вкладка чата "Чат с ботом"

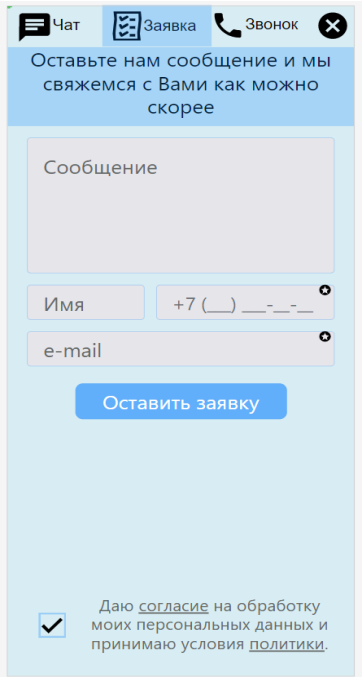


Рис. 3 – вкладка чата "Заявка"

Также обозначим вид начального состояния окна чата на стороннем сайте (см. рис. 4), со страницы которого посетители смогут связаться с операторами. В правом нижнем углу появляется строка “Напишите нам, мы на связи!”, текст которой можно изменять менеджерам. При наведении на надпись всплывают иконки чата и интегрированных приложений.

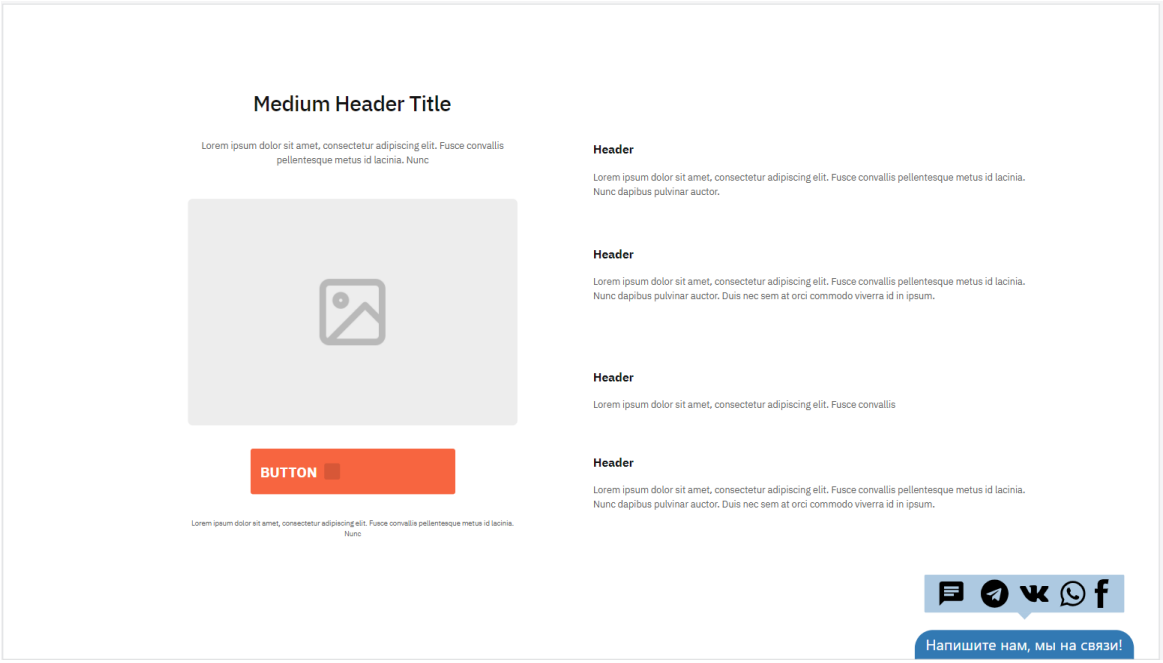


Рис. 4 – вид чата на сайте клиента

Заключение

В результате работы были успешно выполнены поставленные задачи. В дальнейшем можно применить полученные выводы при разработке ИС для онлайн-консультирования, в которой могут заинтересоваться молодые развивающиеся компании, которым необходимо доступное и простое в использовании средство для связи с клиентами.

Библиографический список

1. Перспективы развития науки и образования / Сборник научных трудов по материалам XXX международной научно-практической конференции. Под общей редакцией А.В. Туголукова. 2018. Издательство: Индивидуальный предприниматель Туголуков Александр Валерьевич (Москва).
2. В.А.Карпенко, Разработка чата онлайн консультирования [Электронный ресурс]: бакалаврская работа / – Красноярск.: ФГАОУ ВО «СФУ», 2018. – 54с. – URL: http://elib.sfu-kras.ru/bitstream/handle/2311/74598/diplom_karpenko.pdf?sequence=1&isAllowed=y
3. WebSocket [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/WebSocket (дата обращения: 06.06.2021).
4.

DESIGN AND DOCUMENTATION OF INFORMATION SYSTEM FOR ONLINE-CONSULTING

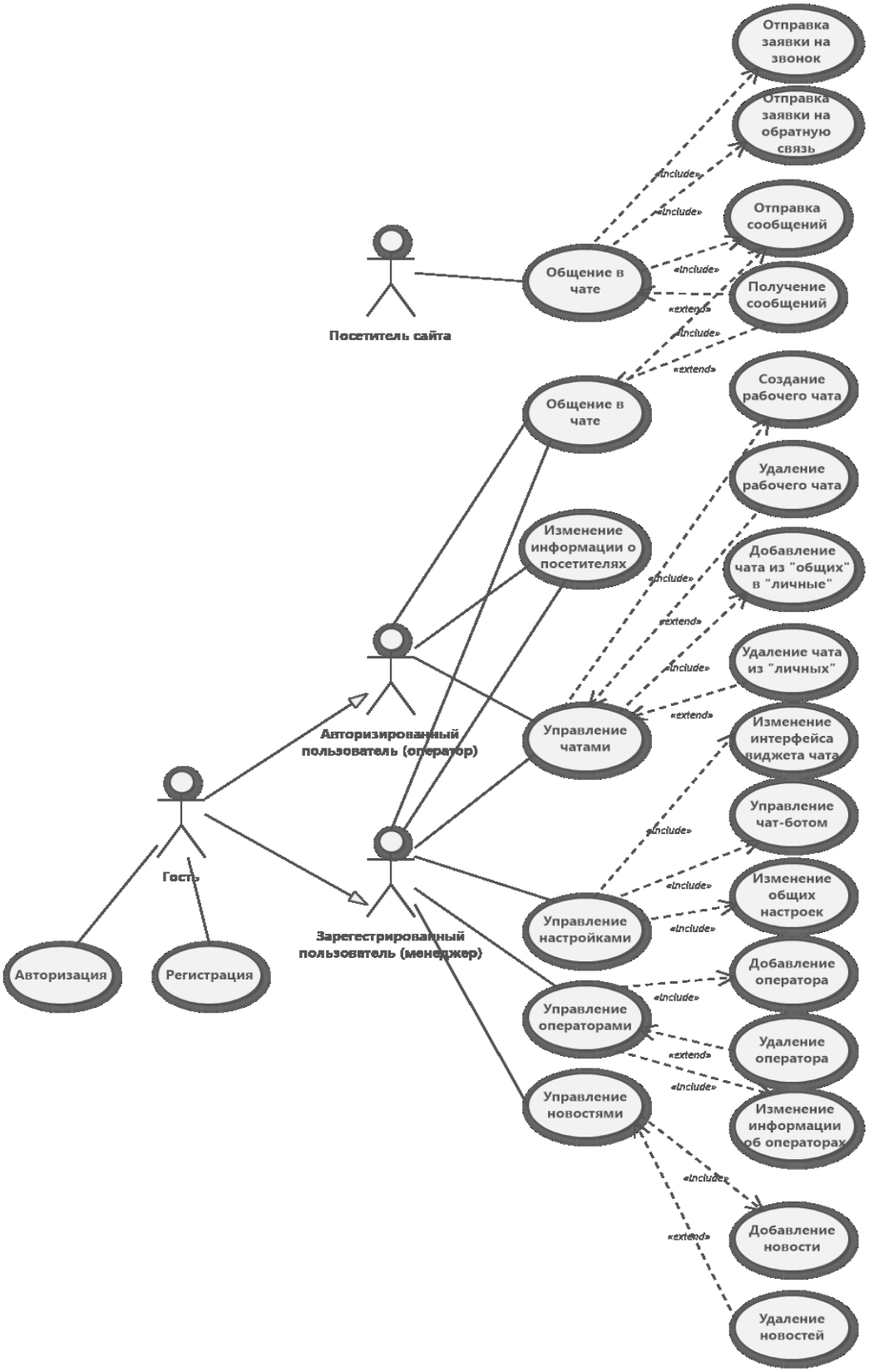
Glavatskikh Kseniya E.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, glavatskikh.kseni@gmail.com

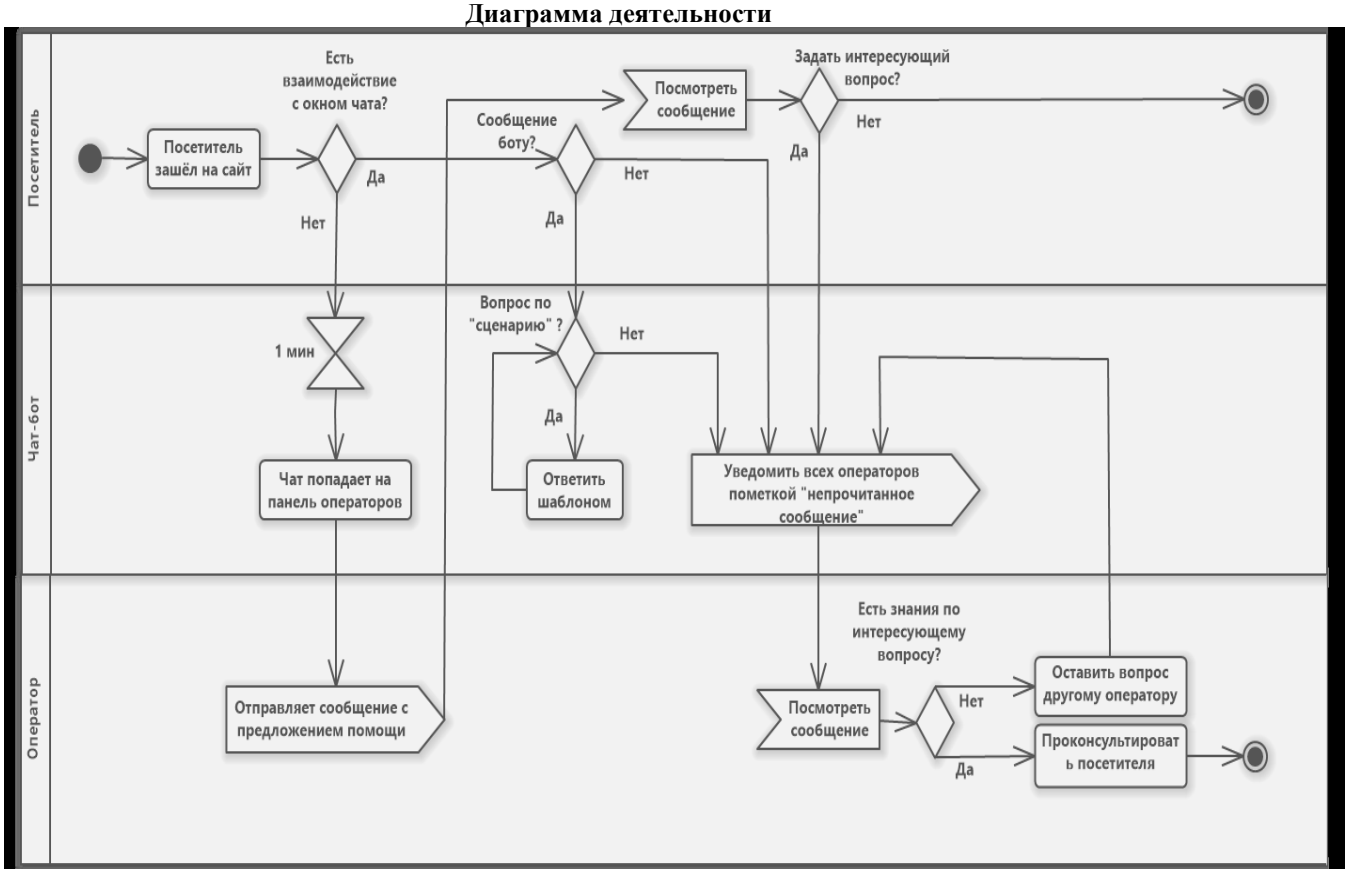
Abstract. This article is devoted to the design and documentation of an information system for online consulting. The relevance of the designed system is determined. The analysis of such information systems is carried out, as a result of which the advantages and disadvantages of each are highlighted. As a result of the analysis, the requirements for the information system, on the basis of which it is designed, are determined. Presented: a brief description of the subject area, an overview of possible tools for designing and modeling information systems. At the design stage, models of the system under development are constructed. For this purpose, a software toolkit is used, combined under the general name CASE-tools. For development, popular implementation tools are considered, and for storing and working with data – DBMS, as a result, suitable for the designed information system are selected. Created: an information system behavior model, structural and physical models, and a user interface layout is presented.

Keywords: information system, online-consulting, chat, chatbot.

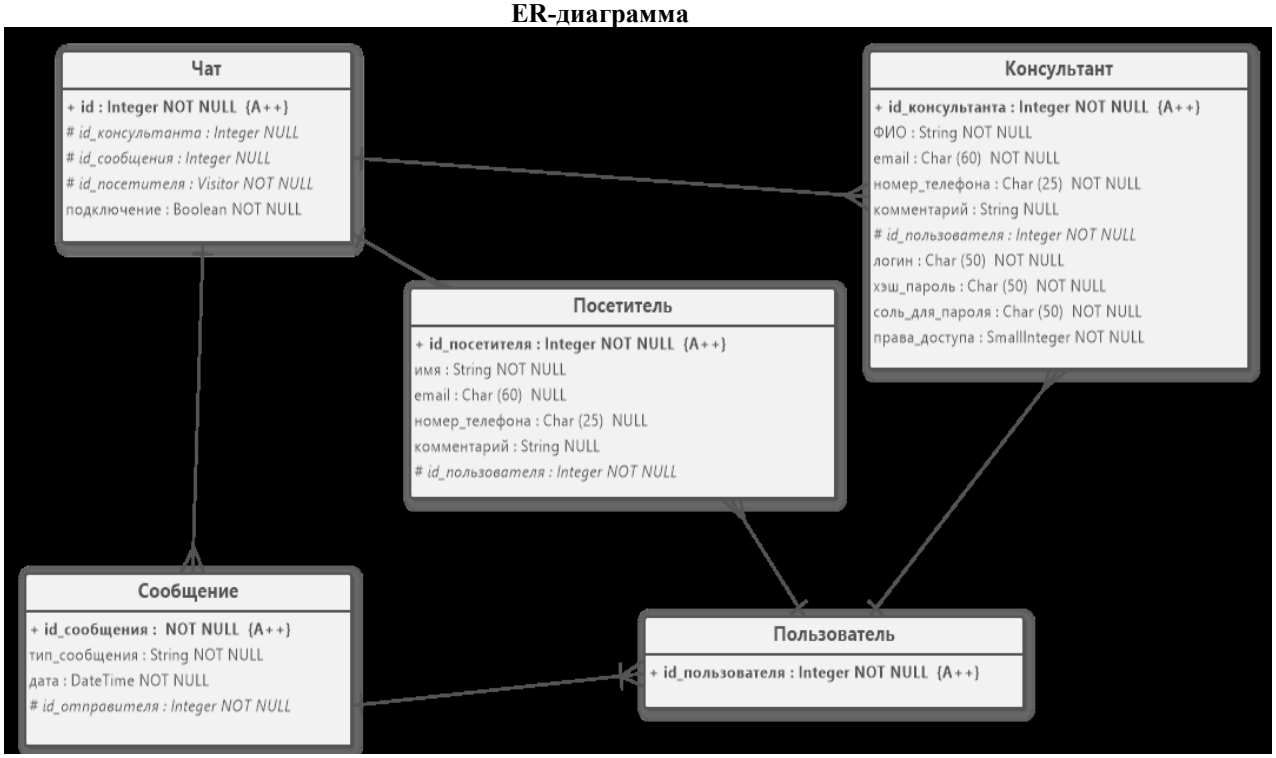
ПРИЛОЖЕНИЕ А.1
Диаграмма прецедентов



ПРИЛОЖЕНИЕ А.2
Диаграмма деятельности



ПРИЛОЖЕНИЕ Б
ER-диаграмма



УДК 004.92

ГРАФИЧЕСКИЙ РЕДАКТОР С ФУНКЦИЕЙ ДЛЯ РАБОТЫ С РЕФЕРЕНСАМИ

Гасанова Наталья Дмитриевна, Анисимова Светлана Игоревна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, netalyges@gmail.ru

Разрабатывается растровый графический редактор со встроенной функцией для работы с референсами и загрузкой изображений из Интернет-ресурса. Рассматриваются и анализируются существующие графические редакторы, их достоинства и недостатки. Показано, что цифровая живопись стремительно развивается и доказана важность создания и совершенствования существующих графических редакторов. Проводится исследование потребностей художников при работе с графическими редакторами, в ходе анализа результатов опроса доказана необходимости реализации встроенной функции для работы с референсами в графическом редакторе. Описано проектирование графического редактора, в том числе его функционал и группы пользователей. Описана разработка графического редактора, в том числе интерфейс и функциональная часть.

Ключевые слова: компьютерная графика, графический редактор, референсы, OpenGL, C#, цифровая живопись.

Сфера изобразительного искусства стремительно развивается с каждым годом, в связи с чем для некоторых людей рисование становится не только хобби, но и профессией. Некоторые художники продолжают использовать традиционные методы рисования, в то время как другие переходят на специализированное программное обеспечение – графические редакторы. Они предназначены не только облегчить работу художников, но и сделать их работы более интересными и оригинальными.

Согласно исследованию Воложаниной Е. А., изложенному в ее статье «Проблематика цифровой живописи» [1], цифровая живопись представляет собой новое направление в искусстве и имеет множество преимуществ над традиционными видами искусства. Важность компьютерной графики и графических редакторов рассматривается и в статье Белозерова О. И. и Селиной А. М. «Цифровая живопись – замена современному искусству?» [2]. Авторы статьи поднимают вопрос потенциала развития цифровой живописи и рассматривают влияние цифровой живописи на современное искусство.

На сегодняшний день в помощь художникам создано множество графических редакторов, но их функционал не может полностью удовлетворить потребности использующих их художников.

Одной из таких потребностей является работа с референсами в графическом редакторе. Референс – это вспомогательное изображение, которое художники используют в своей работе для вдохновения, изучения новых стилей и уточнения того, как правильно передать форму объекта.

Поскольку существующие графические редакторы не предоставляют возможность работать с референсами в самой программе, художникам приходится искать другие способы. Например, некоторые открывают референс на одном из слоев программы, а другие работают одновременно с двумя мониторами, открывая на одном референс, а на другом программу.

Другой потребностью, также относящейся к работе с референсами, является использование изображения, которого может не быть на рабочем устройстве. В таких случаях выходом является поиск изображения в Интернет-ресурсах, скачивание изображения на устройство и открытие его для дальнейшей работы. Подобное решение занимает немало

времени, что в определенных ситуациях может негативно сказаться на результате работы. Также каждое новое изображение занимает дополнительное место в памяти устройства.

К самым популярным и любимым графическим редакторам можно отнести AdobePhotoshop, AdobeIllustrator и PaintToolSAI. AdobePhotoshop является многофункциональным профессиональным редактором изображений, нашедший свое применение в таких областях, как веб-дизайн, рисование, полиграфия, CG-арт и многих других. Одно из его главных особенностей и достоинств является наличие рабочих областей, которые представляют собой специальные интерфейсы, настроенные под выполнение определенных задач. Например, области для работы с 3D-графикой, фотографиями, движением и многим другим. К сожалению, основной интерфейс программы сложен из-за чего порог вхождения довольно высок. Помимо этого, в программе нет встроенной функции для работы с референсами, что весьма неудобно.

AdobeIllustrator предназначен для работы с векторными изображениями. Художники и графические дизайнеры используют программу для создания инфографики, логотипов, схем, иллюстраций и много другого. Многим нравится работать именно в этой программе, т.к. в ней содержится большое количество различных кистей и эффектов, в интернете можно найти множество справочной и обучающей информации в интернете и художник может создания нескольких рабочих областей под свои нужны. Но, как и в AdobePhotoshop, интерфейс программы сложен и нет встроенной функции для работы с референсами.

PaintToolSAI – графический редактор, предназначенный для работы как с векторной, так и с растровой графикой. Художники чаще всего используют программу для создания изображения для аниме, манги и комиксов. У данного графического редактора довольно простой и приятный интерфейс, реализовано распознавание наклона и силы нажатия пера на сенсорной плоскости, что является достоинством программы. Но, как и в двух других ранее рассмотренных графических редакторах, в PaintToolSAI не предусмотрена возможность работать с референсами.

Характеристики и отличительных особенностей разных типов графических редакторов, а также их достоинства и недостатки были рассмотрены также на примере статьи Бухвостова В. О., Козлова А. А., Ноздря О. Д. и Батищева А. В. «В поисках оптимального графического редактора» [3].

Можно сделать вывод, что несмотря на разнообразие существующих графических редакторов есть необходимость улучшать их и создавать новые, более совершенные, которые будут удовлетворять все потребности использующих их художников.

К примеру, графический редактор, при работе с которым не нужно будет придумывать способы для работы с референсами, т.к. в программе будет предусмотрена работа с ними. Также не будет необходимости отдельно искать и загружать на устройство изображения, т.к. будет предусмотрен поиск изображений в самом графическом редакторе.

Для возможности использования изображения в качестве референсов в программе необходим ресурс с бесплатными свободно распространяемыми изображениями. Также у ресурса должен быть свой API и большое количество иллюстраций разных тематик. Среди множества подобных ресурсов был выбран ресурс Pixabay, поскольку он содержит более 2,3 миллиона изображений, в том числе и HD качества, правила использования материалов прозрачны и понятный, а вся необходимая документация для работы с API ресурса находится на самом сайте.

С целью определить потребности художников и понять действительно ли им нужна функция для работы с референсами в графическом редакторе, был проведен опрос. В опросе приняли участие 87 человек разных возрастных категорий, от 12 до 40 лет. В ходе анализа результатов исследования было определено, что более 92% художников используют в своих работах графические редакторы, две трети из них предпочитают растровую графику векторной, а референсы в своих работах используют более 87% опрошенных. Несмотря на то, что большинству респондентов нравится то, как сейчас устроена их работа с референтами, их также как и тех, кому такой формат работы неудобен, заинтересовала возможность

работать с референсами внутри программы. Количество заинтересованных респондентов составило 92%.

Результаты опроса подтверждают значимость графических редакторов, а также разработки встроенной функции для работы с референсами. Следовательно, помимо основных функций, таких как работа со слоями, инструментами, сохранение работы в формате программы и форматах изображений, в разрабатываемом графическом редакторе должна присутствовать функция для работы с референсами и выбора этих референсов как с устройства, так и с ресурса изображений.

Перед разработкой графического редактора было выполнено его проектирование с помощью унифицированного языка моделирования UML. Одним из главных достоинств данного языка моделирования, помимо его простоты и понятности, является то, что это объектно-ориентированный язык, благодаря чему методы описания проектирования семантически близки к методам программирования на современных объектно-ориентированных языках

Поскольку одновременно в программе на одном устройстве может работать лишь один человек и у него должен быть доступ ко всем функциям программы, нет необходимости определять несколько категорий пользователей. Также у каждого пользователя должны быть равные права и возможности, поэтому было принято решение, что будет одна категория пользователей – художник.

Все действия пользователя в программе можно разделить на шесть основных категорий: работа с холстом, работу со слоями, работа с интерфейсом, работа с цветом, работа с референсами и работа с инструментами.

Работа с холстом подразумевает под собой создание новой работы, чистого холста лля рисования, сохранение работы в формате изображений или формате программы, с сохранением структуры слоев, а также открытие изображений и созданных ранее работ.

Работа со слоями включает в себя возможность скрывать слои, в таком случае содержимое слоя не отображается на холсте и рисование на таком слое невозможно. При показе скрытого слоя происходит обратное: все его содержимое отображается на холсте и у пользователя появляется возможность рисовать на данном слое. Также у пользователя должна быть возможность менять слои местами, а точнее меня их уровни. Суть заключается в том, что содержимое слоя меньшего уровня располагается позади содержимого слоя, уровень которого выше выбранного.

Работа с интерфейсом подразумевает под собой возможность изменить цвет темы программы, что не является обязательной функцией графического редактора, но будет полезна пользователям.

Работа с цвет включается в себя выбор двух активных цветов из палитры и возможность изменить их уровень. Основной цвет – это активный цвет первого уровня, которым пользователь рисует в данный момент, дополнительный цвет – запасной цвет, имеющий второй уровень, который пользователь может выбрать в любой момент с помощью смены уровня цвета.

Работа с референсами подразумевает под собой выбор референса с устройства или же поиск и выбор интересующего референса в базе изображений в рамках графического редактора.

Среди базовых инструментов, таких как карандаш, ластик и заливка, должна быть универсальная кисть, текстура которой будет выбираться из файла формата BMP. Сам файл любой пользователь может создать сам, нарисовав нужную текстуру на красном фоне в любом графическом редакторе.

При разработке было принято решение, что графический редактор будет представлять собой клиент-серверное приложение. В качестве программной платформы была выбрана платформа .NET Framework, в качестве языка программирования был выбран объектно-ориентированный язык C#, а для реализации холста и отрисовки объектов – библиотека OpenGL, предназначенная для работы с двухмерной и трехмерной графикой. Данная

библиотека содержит множество полезных и простых функций для отображения, отрисовки и изменения двухмерных объектов, что подробно рассмотрено в статье Ковтунова Д. С. «Обзор спецификации OpenGL» [4]. Для реализации взаимодействия с ресурсом свободно распространяемых изображений Pixabay по средством API данного ресурса была выбрана библиотека PixabaySharp.

Поскольку графический редактор будет представлять собой клиент-серверное приложение, он будет состоять из двух основных частей: клиентская часть – представляет собой интерфейс, с которым напрямую взаимодействует пользователь во время работы, а также серверная часть – часть графического редактора, отвечающая за выполнения команд и запросов, поступающих с клиентской части программы.

При реализации клиентской части был разработан интерфейс программы (см. рис.), состоящий из пяти основных блоков: меню, блок для работы с референсами, холст, блок для работы с инструментами и блок для работы со слоями.

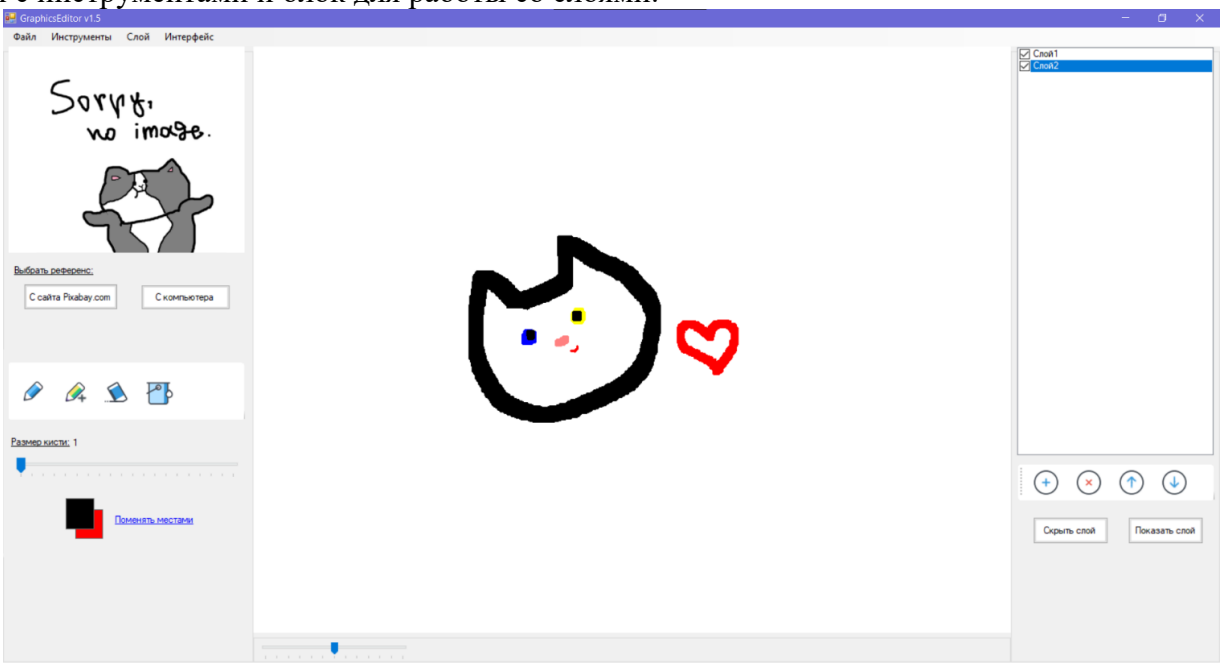


Рис. Интерфейс программы

Все функции меню которого разделены на группы и расположены в соответствующих вкладках. Блок для работы с референсами содержит область загрузки изображения и две кнопки для выбора способа загрузки. Блок с инструментами состоит из трех подблоков: набор инструментов рисования, инструмент изменения размера кистей и ластика, а также подблок для работы с цветом. Самым крупным блоком рабочей области графического редактора является холст, под которым находится инструмент для масштабирования содержимого слоев. Последний блок, занимающий всю правую часть рабочей области – блок для работы со слоями. Он содержит список слоев и набор кнопок для управления ими., в том числе для создания и удаления слоев, изменения их уровня и видимости.

При реализации серверной части графического редактора были разработаны три класса: anBrush, anLayer и anEngine.

Класс anBrush, реализует основу для объекта «Кисть». Другие инструменты графического редактора реализуются за счет различных методов и изменения значений атрибутов данного класса.

Класс anLayer определяет слои холста графического редактора. Именно на слоях происходит обрисовка объектов кистью, реализуемой классом anBrush.

anEngine – класс, представляющий собой холст графического редактора. Он является основным, т.к. при запуске программы в первую очередь создается экземпляр холста, после чего пользователь уже может работать со слоями, инструментами, изображениями и файлом

программы. Также в данном классе создаются экземпляры классов anBrush и anLayer для дальнейшей работы

Помимо классов был разработан ряд функций для взаимодействия с ресурсом Pixabay, работы с цветом, изображениями, инструментами, слоями, отрисовкой и изменением объектов и многим другим.

С целью оптимизировать работу графического редактора были изучены алгоритмы оптимизации создания и изменения объектов в уже существующих графических редакторах, описанные в статье Степанова А.А. «Сравнительный анализ работы алгоритма оптимизации в графических редакторах» [5].

Результатом работы стал растровый графический редактор, в котором реализованы не только такие базовые функции, как выбор и настройка инструментов рисования, взаимодействие со слоями и сохранение работы в различных форматах, но и функция для работы с референсами в рамках графического редактора, а таже выбор референса из базы изображений ресурса Pixabay.

Библиографический список

5. *Воложанина Е. А*. Проблематика цифровой живописи [Электронный ресурс] URL: https://cyberleninka.ru/article/n/problematika-tsifrovoy-zhivopisi/viewer (дата обращения: 10.06.2021).
6. *Белозеров О. И., Селина А. М*. Цифровая живопись – замена современному искусству? [Электронный ресурс] URL: https://cyberleninka.ru/article/n/tsifrovaya-zhivopis-zamena-sovremennomu-iskusstvu/viewer (дата обращения: 10.06.2021).
7. *Бухвостов В. О., Козлов А. А., Ноздря О. Д., Батищев А. В*. В поисках оптимального графического редактора [Электронный ресурс] URL: https://cyberleninka.ru/article/n/v-poiskah-optimalnogo-graficheskogo-redaktora/viewer (дата обращения: 10.06.2021).
8. *Ковтунов Д. С*. Обзор спецификации OpenGL [Электронный ресурс] URL: https://cyberleninka.ru/article/n/obzor-spetsifikatsii-opengl/viewer (дата обращения: 10.06.2021).
9. *Степанов А. А*. Сравнительный анализ работы алгоритма оптимизации в графических редакторах [Электронный ресурс] URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-raboty-algoritma-optimizatsii-v-graficheskih-redaktorah/viewer (дата обращения: 10.06.2021).

GRAPHIC EDITOR WITH A FUNCTION FOR WORKING WITH REFERENCES

Gasanova Nataly D., Anisimova Svetlana I.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, netalyges@gmail.ru

Abstract. Development of a raster graphics editor with a built-in function for working with references and downloading images from an Internet resource. The existing graphic editors, their advantages and disadvantages are considered and analyzed. It is shown that digital painting is rapidly developing, and the importance of creating and improving existing graphic editors is proved. A study was conducted on the needs of artists when working with graphic editors. The analysis of the research results proves the need to implement a built-in function for working with references in a graphical editor. It describes the modeling of a graphic editor, including its functionality and user groups. It describes the development of a graphic editor, including the interface and the functional part.

Keywords: computer graphics, graphic editor, references, OpenGL, C#, digital painting.

УДК 004.432

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ФРЕЙМВОРКИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ WEB-РАЗРАБОТКИ»

Кулижский Никита Сергеевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, kulijsky@yandex.ru

В статье рассмотрена система, которая предназначена для получения актуальной информации о фреймворках языка программирования Python для web-разработки. Проведен анализ фреймворков, которые популярны на данный момент. Обоснована актуальность проектирования информационной системы. Построены диаграмма прецедентов, диаграмма активности, создан прототип интерфейса.

Ключевые слова: фреймворки, Python, web-разработка, backend.

Введение

На данный момент с развитием сетевых технологий всемирная компьютерная сеть стала употребляться в очень многих областях жизни каждого человека. С каждым годом количество людей, пользующихся интернетом, постоянно растет. В связи с этим, появилось огромное количество технологий, которые предоставляют возможность создания и разработки web-сайтов и web-приложений.

Одной из известных технологий является Python. Как правило, при работе в области web-программирования для Python используются различные фреймворки. Главной целью их использования является решение однотипности задач при каждой разработке. Harry J.W. Percival, в своей книге про web-разработку на языке Python, пишет о том, что количество людей, разрабатывающих web-сервисы на Python, постоянно увеличивается.

Актуальность данной темы состоит в том, что существует огромное количество фреймворков языка программирования Python, и из-за этого пользователь теряется в выборе необходимой программной платформы.

Цель данной работы состоит в проектировании информационной системы по описанию различных фреймворков для web-разработки на языке Python

Анализ существующих фреймворков

Рассмотрим фреймворки, которые чаще всего используются в web-разработке в настоящий момент:

Django – самый популярный фреймворк языка программирования Python для web-разработки. Поддерживается организацией Django Software Foundation и предоставляется с открытым исходным кодом. Django описывают как «веб-фреймворк для перфекционистов с дедлайнами». Его создали, чтобы переходить от прототипов к готовым сервисам как можно быстрее.

Flask – это микрофреймворк языка программирования Python с открытым исходным кодом. Является одним из самых популярных наряду с Django. Разработан в 2010 году австрийским программистом Армином Ронахером.

Tornado – это веб-фреймворк Python и асинхронная сетевая библиотека, первоначально разработанные FriendFeed в 2009 году. Используя неблокирующий сетевой ввод-вывод, Tornado может масштабироваться до десятков тысяч открытых подключений, что делает его идеальным для технологии push, веб-сокетов и других приложений, требующих длительного соединения с каждым пользователем.

Bottle – легкий фреймворк, который помещается всего в один файл. Разработан в 2009 году разработчиком Марселем Хеллкампом и распространяется с открытым исходным кодом. Несмотря на легкость, он предоставляет широкие возможности, которых хватает для всех проектов, кроме очень больших. Он предлагает отправку запросов (маршруты) с поддержкой параметров URL, шаблоны, встроенный HTTP-сервер и адаптеры для многих сторонних WSGI / HTTP-серверов и механизмов шаблонов.

Pyramid – Python фреймворк, предназначенный для программирования трудных объектов и решения многофункциональных задач. Разработан в 2010 году разработчиками Беном Бангертом и Джеймсом Гарднером. На создание Pyramid наибольшее влияние оказали такие фреймворки, как Django, Zope.

Web2Py – масштабируемый фреймворк Python, который имеет собственный IDEwhich, в который входит редактор кода, debugger, deployer. Разработан Massimo De Pierro в 2007 году. Изначально был придуман как учебный инструмент с акцентом на usability. Архитектура была разработана под влиянием таких фреймворков, как RoR и Django.

CherryPy – это web-фреймворк Python, который предоставляет дружественный интерфейс для HTTP-протокола для разработчиков Python. Его также называют библиотекой веб-приложения. Разработан командой CherryPy под руководством Реми Делона в 2002 году.

TurboGears – фреймворк с открытым исходным кодом для быстрого создания приложений, которые работают с данными. Был разработан в 2005 году Кевином Дангуром и Марком Раммом.

Описанные выше фреймворки отлично зарекомендовали себя в различных проектах и используются в настоящее время. Однако некоторые программные платформы не предназначены для серьезных и больших проектов из-за своей легковесности.

Проектирование информационной системы

Для успешной реализации информационной системы должно присутствовать описание ее функционала. Так были построены диаграммы прецедентов для разных типов пользователей: гостя (см. рисунок 1) и авторизованного пользователя (см. рисунок 2).

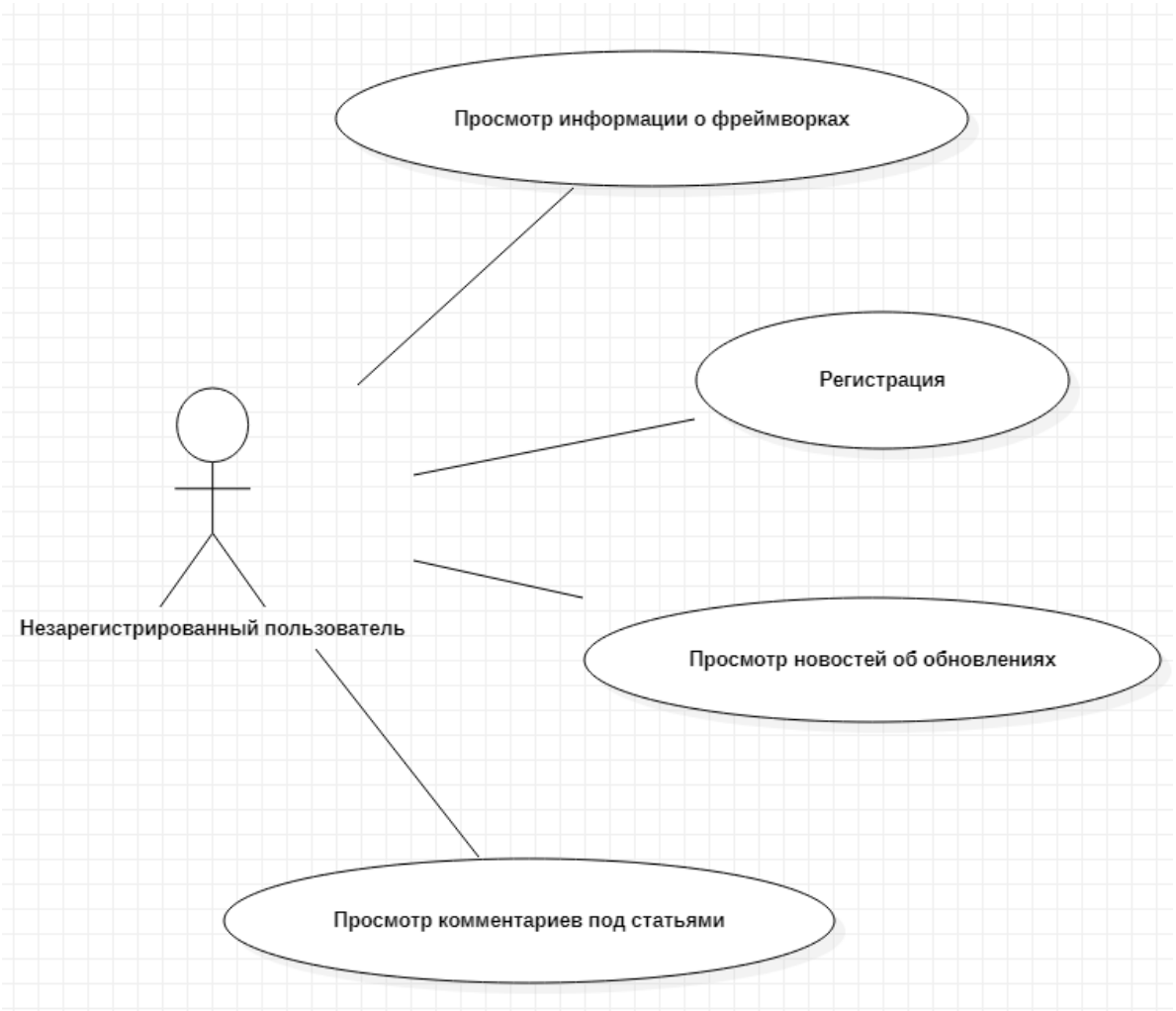


Рис. 1 – Диаграмма прецедентов для гостя

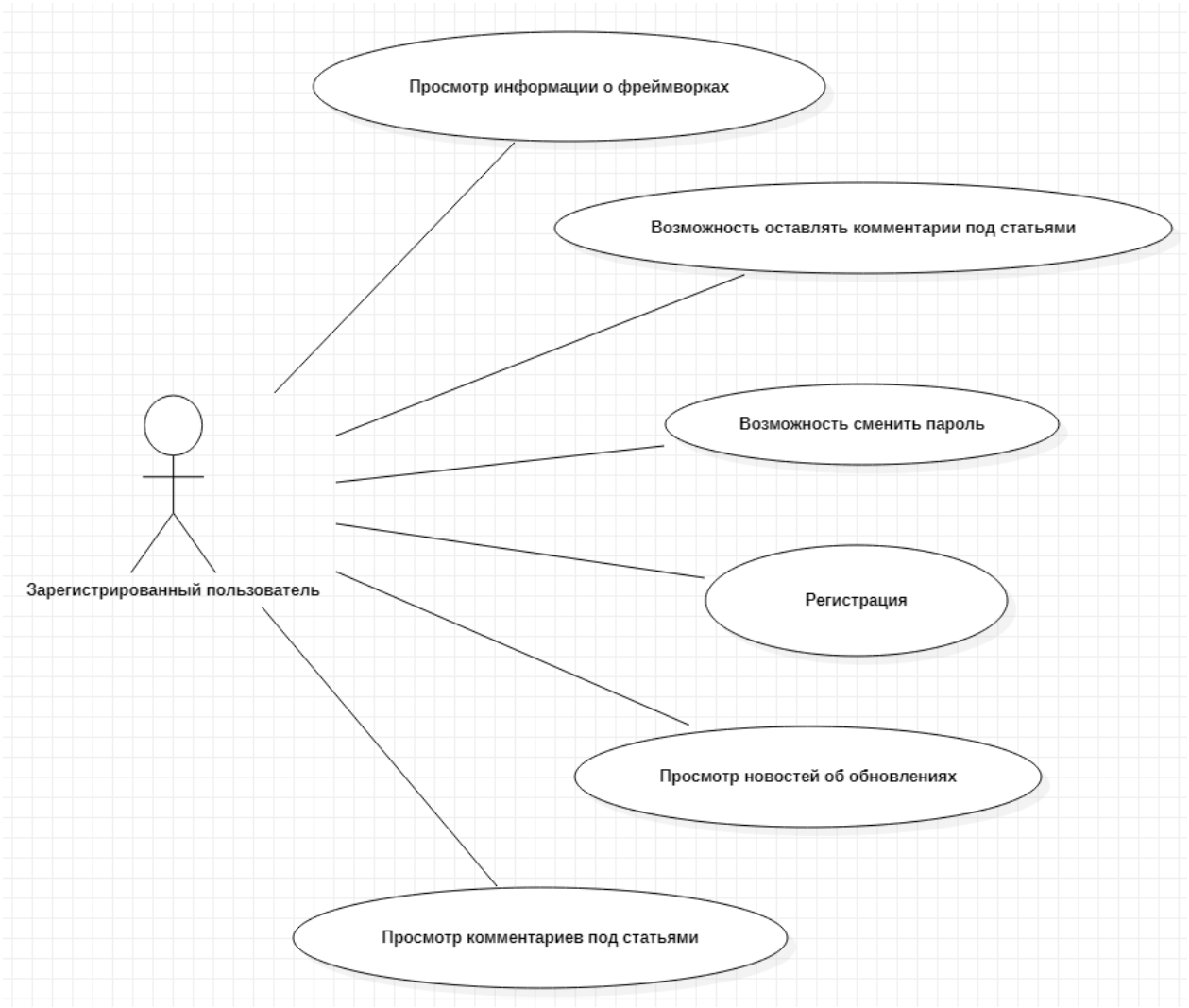


Рис. 2 – Диаграмма прецедентов для авторизованного пользователя

Незарегистрированный (неавторизованный) пользователь имеет доступ к просмотру информации о фреймворках, новостей об обновлениях. А также может зарегистрироваться и прочитать комментарии под каждым из фреймворков.

Зарегистрированный (авторизованный) пользователь также имеет доступ к просмотру всей информации на сайте, и, в дополнение, может оставлять комментарии под каждой из статей. Еще он имеет возможность поменять пароль. Авторизация происходит по логину и паролю.

Авторизация или регистрация пользователя происходит с взаимодействием интерфейса web-сайта. Для процедуры авторизации построена диаграмма активности (см. рисунок 3).

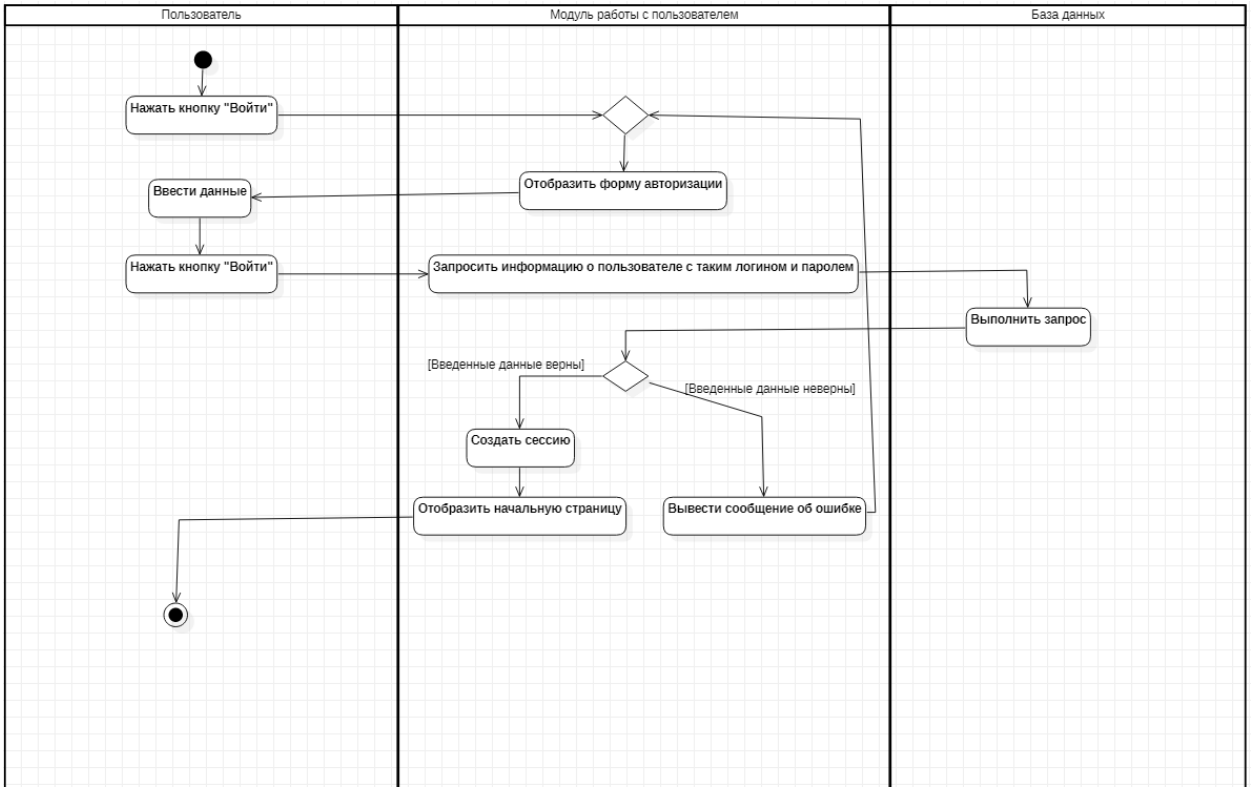


Рис. 3 – Диаграмма активности для процедуры авторизации

Для процедуры регистрации построена аналогичная диаграмма активности (см. рисунок 4).

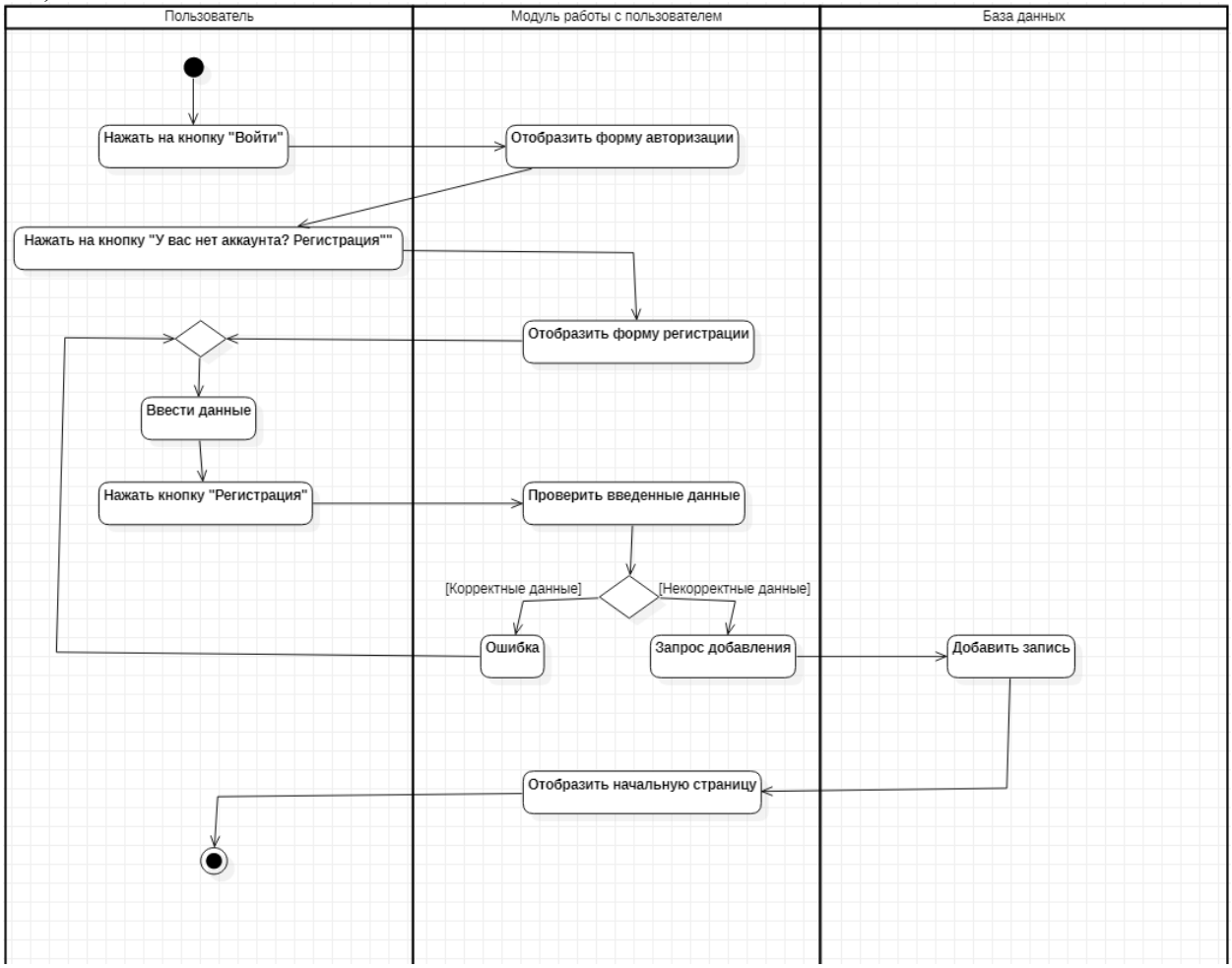


Рис. 4 – Диаграмма активности для процедуры регистрации

Для удобства работы с разрабатываемой системой, требуется спроектировать понятный пользовательский интерфейс. Пользовательский интерфейс – это внешний вид продукта, способ общения между пользователем и программой.

Разработанный макет интерфейса интуитивно понятен обычному пользователю, а также лёгок в эксплуатации (см. рисунок 5).

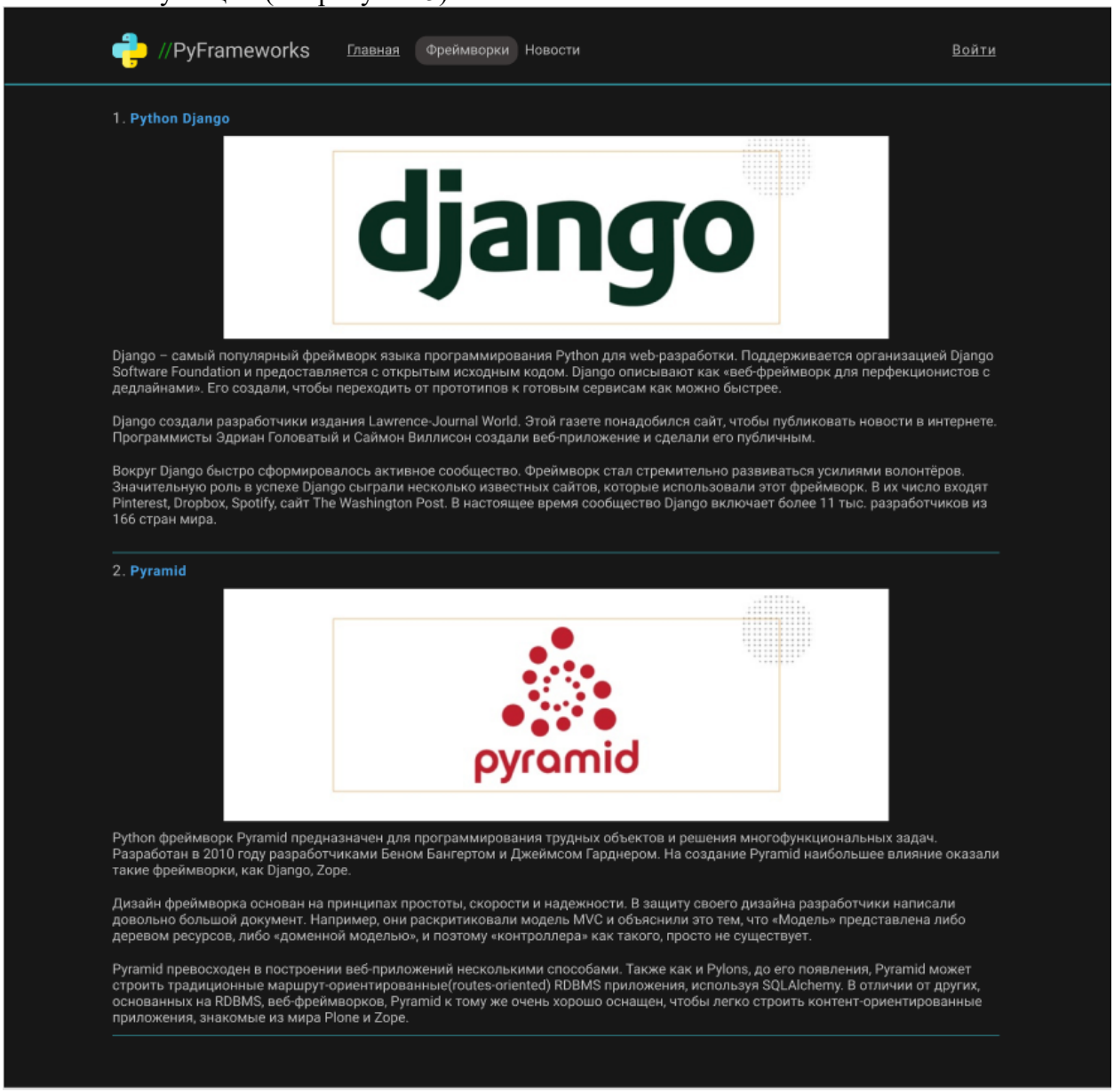


Рис. 5 – Прототип интерфейса главной страницы сайта

Заключение

Разработанная модель информационной системы позволяет получить информацию о различных фреймворках языка программирования Python. А также обсудить нововведения по конкретному фреймворку с другими пользователями.

Как итог система позволяет пользователю значительно сократить время на выбор программной платформы для своего проекта.

Библиографический список

1. Harry J.W. Percival Test-Driven Development with Python. // Science, technology and education. – 2017. – С. 558-560.
2. Васильев П.А. Web-программирование на языке Python. Фреймворки Django, Flask // Наука, техника и образование. – 2016. – С. 38–39.
3. Пастушенко, В. А. Обзор web-фреймворков языка Python / В. А. Пастушенко, А. И. Лаптева // Современные тенденции развития фундаментальных и прикладных наук: Материалы Всероссийской с международным участием научно-практической конференции, Брянск, 12 марта 2018 года / Под ред. С.А. Коньшаковой. – Брянск: Федеральное государственное бюджетное образовательное учреждение высшего образования "Брянский государственный инженерно-технологический университет", 2018. – С. 54-58.

DEVELOPMENT AND DOCUMENTATION THE INFORMATION SYSTEM «PYTHON PROGRAMMING LANGUAGE FRAMEWORKS FOR WEB DEVELOPMENT»

Kulizhskii Nikita S.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, kulijsky@yandex.ru

The article describes a system that is designed to get up-to-date information about the Python programming language frameworks for web development. The analysis of the frameworks that are popular at the moment is carried out. The relevance of the information system design is justified. A use case diagram and an activity diagram were built, and an interface prototype was created.

Keywords: frameworks, python, web development, backend.

УДК 004.41

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ФРЕЙМВОРКИ ДЛЯ РЕШЕНИЯ ЗАДАЧ ИЗ ОБЛАСТИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И МАШИННОГО ОБУЧЕНИЯ»

Угринов Валерий Александрович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, valeraugalval@gmail.com

В статье поэтапно разобран процесс разработки информационной системы по описанию различных фреймворков, предназначенных для решения задач из области искусственного интеллекта и машинного интерфейса. Определён предмет и объект исследования, сформулирована актуальность разработки данной информационной системы, а также поставлены задачи, которые необходимо решить в ходе проектирования. Проведён краткий анализ существующих фреймворков, которые чаще всего используются в соревнованиях по машинному и глубокому обучению на платформе Kaggle, определены задачи, которые должны обеспечиваться разрабатываемой информационной системой. Выбраны средства проектирования и разработки. Представлена модель данных и модель предметной области. Модель данных состоит из диаграммы последовательности и активности, а модель

предметной области из диаграммы сущность – связь. После чего был разработан пользовательский интерфейс, включающий в себя прототип страницы с информацией о фреймворке Keras.

Ключевые слова: искусственный интеллект, машинное обучение, веб-сайт.

За последние несколько лет тема искусственного интеллекта вызвала большой ажиотаж. В наши дни проблема стремительного развития ИИ (искусственный интеллект) не только в деятельности крупных компаний, но и в повседневной жизни продолжает звучать особенно актуально. Это обусловлено тем, что внедрение в деятельность компаний искусственного интеллекта позволяет частично заменить человеческий труд, соответственно сократить оплату труда работников, а значит получить экономическую выгоду. Искусственный интеллект позволяет компьютерам выполнять ряд задач, которые раньше были под силу только человеку. Крупные компании пытаются ввести технологии ИИ и машинного обучения в свой бизнес. Появляются всё более технологичные машины с автопилотом, виртуальные помощники, собеседники. Так Франсуа Шолле, разработчик фреймворка Keras, в своей книге повествует о том, что за прошедшие пять лет количество людей, работающих над машинным обучением, увеличилось с несколько сотен до нескольких десятков тысяч, а также, что машинное обучение заняло центральное место среди стратегических продуктов технологических гигантов, таких как Google, Facebook, Baidu, Microsoft и Intel [1].

В связи с актуальностью темы ИИ и машинного обучения появилась потребность в разработке информационной системы по описанию различных фреймворков, предназначенных для решения задач по ИИ и машинному обучению.

Таким образом, целью данной работы является проектирование информационной системы по описанию различных фреймворков для решения задач по ИИ и машинному обучению.

Объект исследования – фреймворки для решения задач из области ИИ и машинного обучения.

Предмет исследования – проектирование информационной системы по описанию фреймворков для решения задач из области ИИ и машинного обучения.

Задачи, которые необходимо решить в ходе проектирования информационной системы:

1) изучение и анализ существующих фреймворков для решения задач из области искусственного интеллекта и машинного обучения;
2) обзор и выбор средств проектирования информационной системы;
3) проектирование информационной системы;
4) разработка пользовательского интерфейса.

В работе представлен анализ предметной области, разобраны популярные фреймворки для решения задач из области искусственного интеллекта и машинного обучения.

Разобраны такие фреймворки как:

— tensorflow;
— pytorch;
— theano;
— keras;
— caffe;
— mxnet;
— cntk;
— deeplearning4j.

Так TensorFlow обладает отличным средством мониторинга процесса обучения моделей и визуализации, что помогает быстро обнаруживать ошибки, но модель

архитектуры данного фреймворка низкоуровневая, что означает большой порог вхождения, а PyTorch не имеет встроенного средства визуализации, но использует динамический граф. Theano на данный момент не развивается. Caffe интегрируется с MATLAB, поддерживает архитектуру CUDA и решает широкий спектр задач. Keras является надстройкой на бэкэндами TensorFlow, Theano и предназначен для небольших наборов данных. MXNet имеет поддержку множества языков программирования, таких как C++, Python, JavaScript, Matlab, Go, R, Perl и Scala. У CNTK ограничена поддержка архитектуры ARM. A Deeplearning4j предназначен для языка программирования java.

На основе анализа существующих фреймворков, которые используются в соревнованиях по машинному и глубокому обучению были сформулированы задачи, которые должны обеспечиваться разрабатываемой информационной системой:

— предоставлять описание фреймворка;
— предоставлять возможности фреймворка;
— предоставлять описание решаемых задач;
— предоставлять интерфейс для пользователя.

Важной частью проектирования информационной системы является выбор средств проектирования и разработки. Были выбраны такие средства:

— dia – проектирование ИС;
— html5, css, javascript – пользовательский интерфейс;
— php – внутренняя часть веб-сайта;
— mysql – хранение и работа с данными;
— mysql workbench – проектирование БД (база данных).

При моделировании информационной системы были составлены модели разных видов: модель предметной области, модель данных информационной системы.

В модели данных были разработаны диаграмма последовательностей, диаграмма активностей.

В диаграмме последовательности был разобран процесс добавления комментария на сайт, в ней есть связь между зарегистрированным пользователем, интерфейсом, API и базой данных (см. рис. 1).

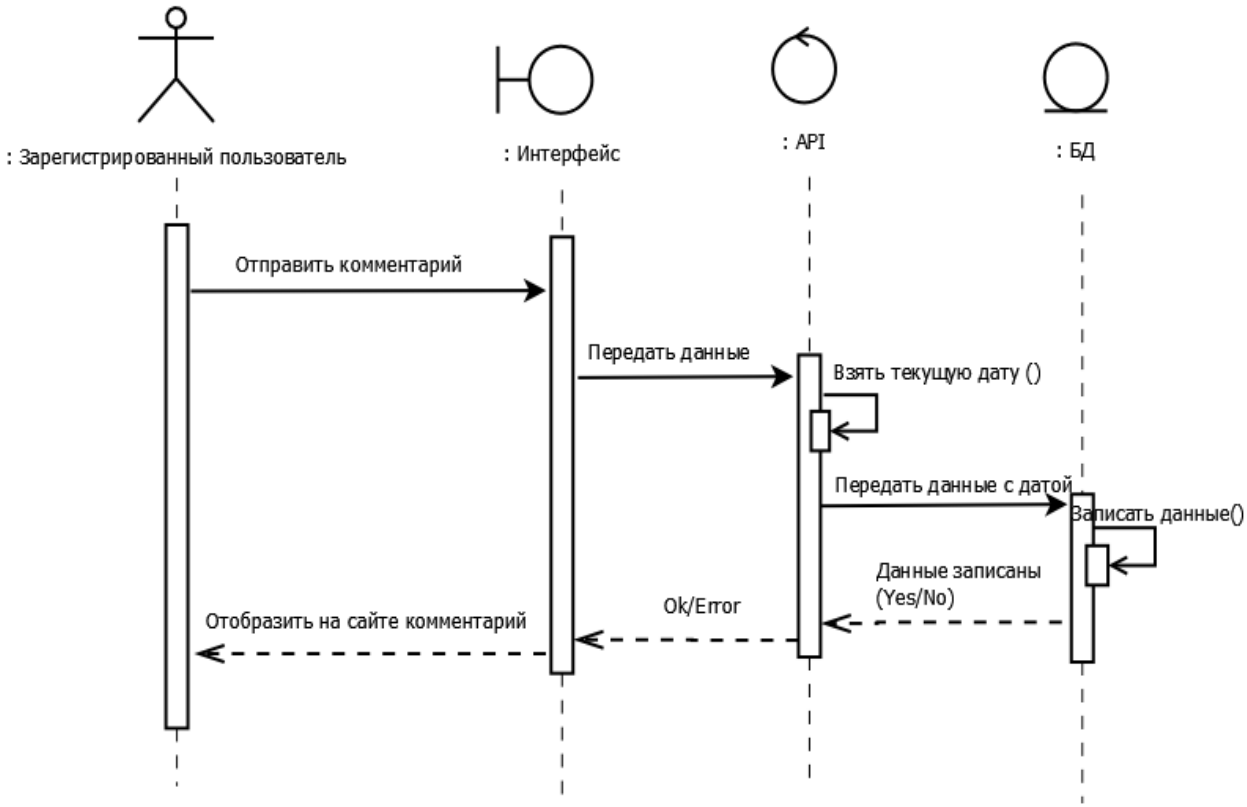


Рис. 1 Диаграмма последовательностей

В диаграмме активности был разобран процесс регистрации и авторизации (см. рис. 2).

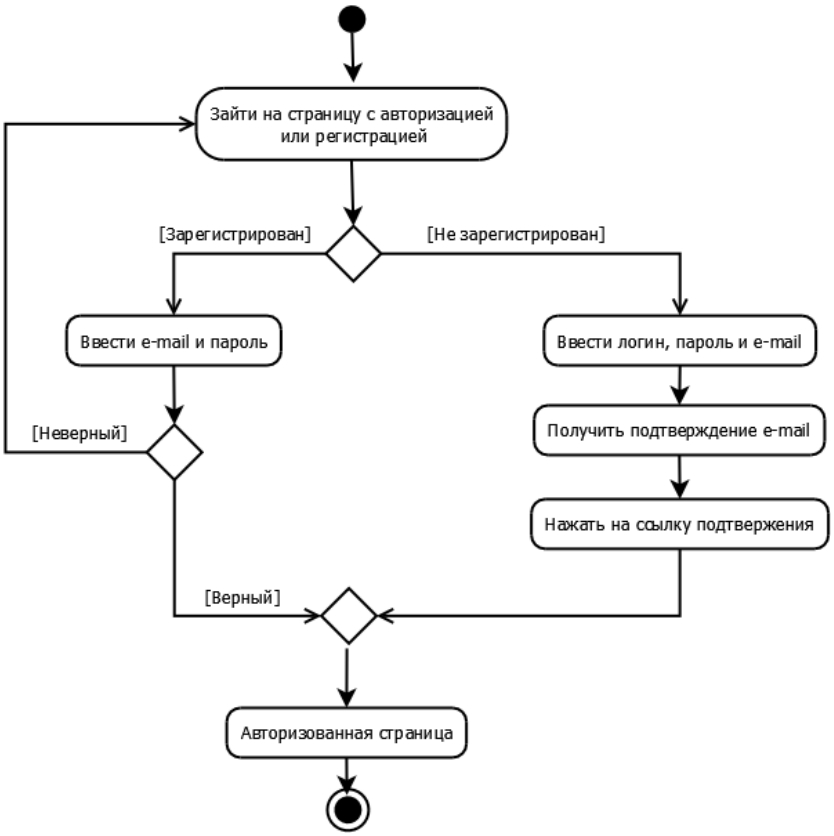


Рис. 2 Диаграмма активности

При проектировании концептуальной модели предметной области была разработана диаграмма сущность-связь, в которой описаны сущности, определяющие информационную систему. Сущность “Зарегистрированный пользователь”, Сущность “Комментарии”, Сущность “Статьи”, Сущность “Родительские и дочерние комментарии”. Реализация диаграммы сущность – связь была разработана в программе MySQL Workbench (см. рис. 3).

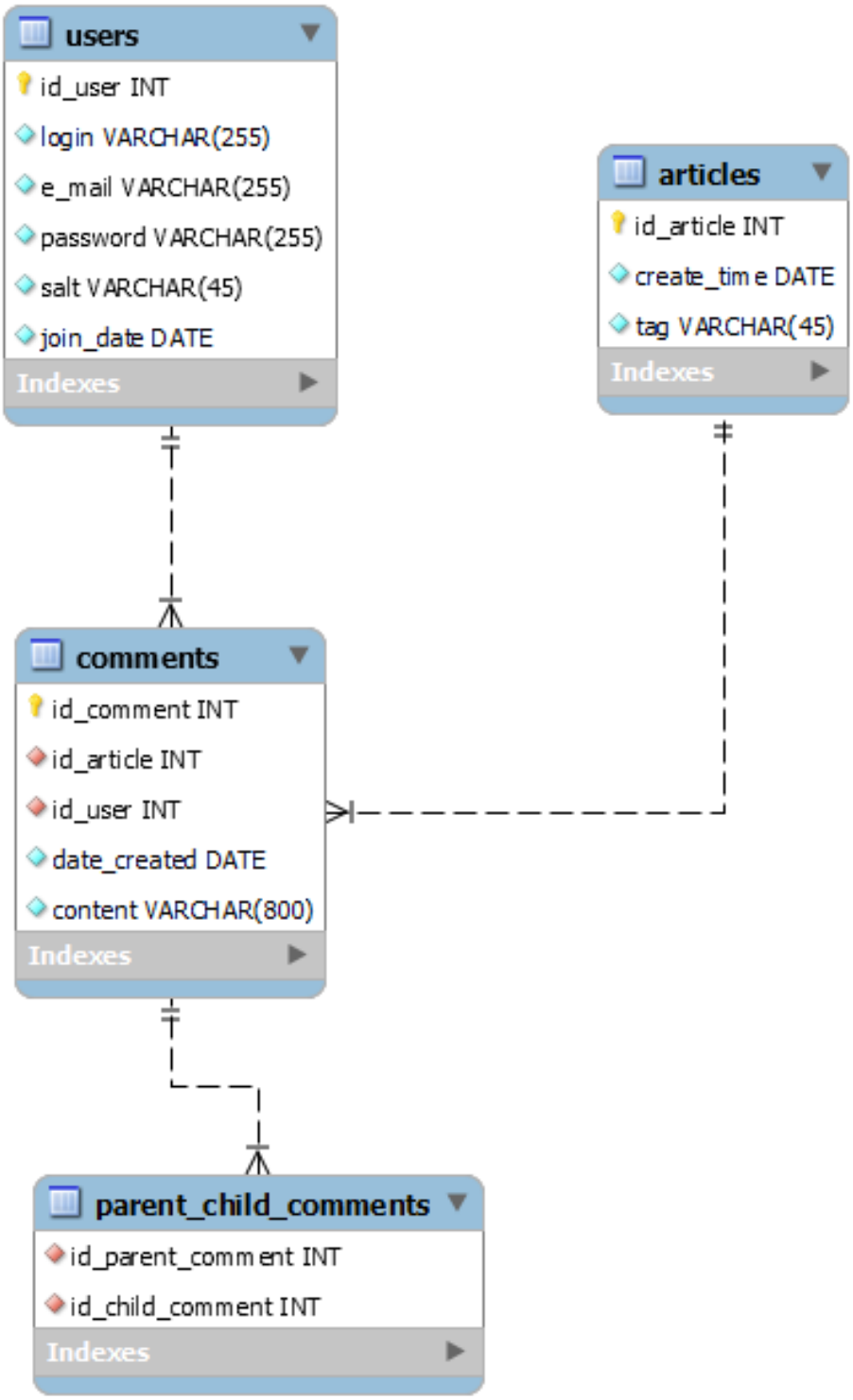


Рис. 3 Диаграмма “Сущность – связь”

Исходя из предыдущих этапов моделирования был разработан пользовательский интерфейс информационной системы. Для этого был использован бесплатный онлайн-инструмент для создания пользовательского интерфейса – Figma.

Так, был разработан прототип страницы с информацией о фреймворке Keras (см. рис. 4).

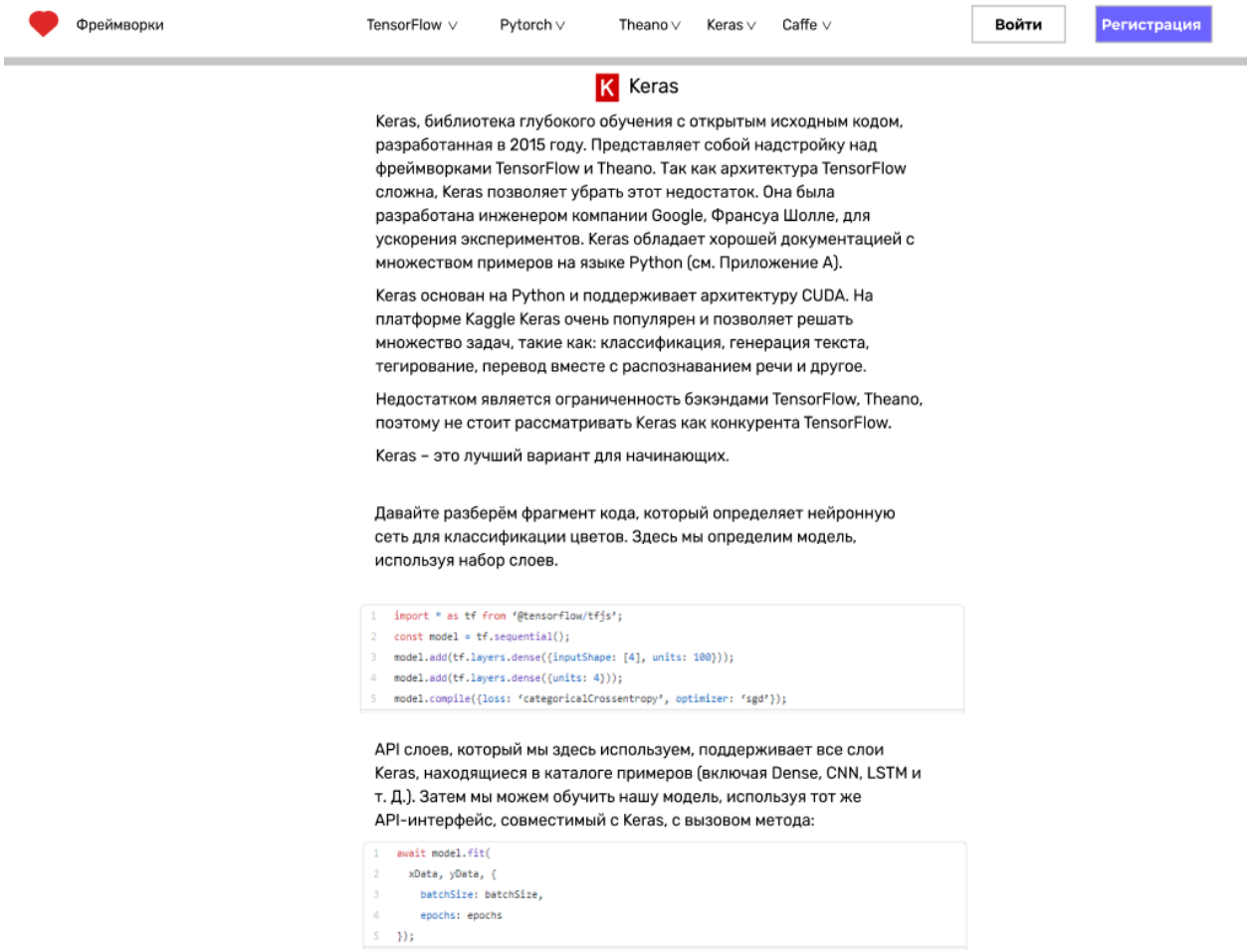


Рис. 4 Прототип страницы с информацией о фреймворке Keras

В дальнейшем развитии данной информационной системы планируется расширение функционала, среди которого:

— введение тестовой системы оценки знаний по выбранным фреймворкам;

— введение интерактивного обучения;

— добавить новую роль, у которой была бы функция добавления новых статей на сайт и редактирование имеющихся.

Библиографический список

1. *Шолле Ф.* Глубокое обучение на Python. – СПБ.: Питер, 2020. – С. 25-48.

DESIGNING THE INFORMATION SYSTEM "FRAMEWORKS FOR SOLVING PROBLEMS IN THE FIELD OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING"

Ugrinov Valeriy A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, valeraugalval@gmail.com

The article gradually analyzes the process of developing an information system by describing various frameworks designed to solve problems in the field of artificial intelligence and machine interface. The subject and object of research are defined, the relevance of the development of this information system is formulated, and the tasks that need to be solved during the design are set. A brief analysis of the existing frameworks is carried out, the tasks that should be provided by the developed information system are defined. Design and development tools are selected. The data model and the domain model are presented. The data model consists of a use case diagram for two user roles, a sequence and activity diagram, and the domain model consists of an entity – relationship diagram. After that, the user interface was developed, including a registration form, a block with comments, the top of the page, and a prototype page with information about the Keras framework.

Keywords: artificial intelligence, machine learning, website.

УДК 004.055

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ВЫЕЗДНОЙ АВТОМОЙКИ

Суханов Иван Дмитриевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, vanekk2000@gmail.com

Рассматриваются основные средства для проектирования и документирования информационной системы для выздной автомойки. В ходе исследования были проанализированы: аудитория интернета, рынок мобильных приложений за 2020 год и приложения аналоги. Выявлено, что за 2020 год среднесуточная аудитория интернета в России составила 65% от населения страны [1]. Данная статистика показывает, что люди в России начинают все больше использовать смартфоны и планшеты нежели персональные компьютеры. А это значит, что больше используются мобильные приложения. Статистика показывает, что за период ограничительных мер из-за «Covid-19» рынок мобильных приложений, оказывающих услуги разного рода, увеличился на 199% [2]. В результате анализа аналогов информационных систем для выездных автомоек, были представлены основные требования к нашей информационной системе: комфортный в использовании интерфейс, возможность добавления нескольких автомобилей (согласно статистике [3] для некоторых людей – это актуальная функция), функцию автоматического определения местоположения и построения оптимальных маршрутов для персонала, удобную форму заполнения заявки на мойку, систему скидок для клиентов и систему мотивации для персонала. В результате анализа средств проектирования и разработки было выбрано инструментальное программное обеспечение проектирования информационной системы для выездной автомойки: CASE-средство StarUML для моделирования системы [4], [5]; веб-

сервис Figma.ru для разработки прототипа интерфейса приложения для клиентов и персонала. В результате проектирования информационной системы для выездной автомойки мы описали поведение системы и взаимодействие с ней пользователя как со стороны клиентов, так и со стороны персонала. Для описания взаимодействий клиента или персонала с системой использовали: диаграммы прецедентов, диаграммы деятельности, диаграммы последовательности и описали с помощью блок-схемы алгоритм приема заказов. На основе построенных диаграмм мы смогли спроектировать возможный интерфейс системы, который содержит функционал, описанный в диаграммах.

Ключевые слова: средства для проектирования и документирования, выездная автомойка, дружественный интерфейс, проектирование мобильного приложения, инструментальное программное обеспечение, CASE-средство, приложение для клиентов, приложение для персонала, диаграмма прецедентов, диаграмма последовательности, блок-схема, проектирование интерфейса.

Библиографический список

1. Анализ рекламы: сравнительный анализ рынка рекламы в России [Электронный ресурс] // Mediascope. URL: https://mediascope.net/news/1250827 (дата обращения: 22.05.2021);
2. За первый квартал 2020 года прирост новых интернет-магазинов составил порядка 199% // Исследование «Как поменялся рынок электронной коммерции в связи с коронавирусом» [Электронный ресурс] // AdvantShop. URL: https://www.advantshop.net/blog/common/analitika-po-onlain-prodazham-v-period-karantina?fbclid=IwAR2kEirS64VAStKVl-jkbzxbAeW-5i95h4sNtzh-XTLWRwxYr77smt1tXYw (дата обращения: 22.05.2021);
3. У каждой шестой семьи в России – 2 и более автомобиля [Электронный ресурс] // Автостат. URL: https://www.autostat.ru/news/22291 (дата обращения 24.05.2021);
4. Автоматизация СМК: CASE-средства [Электронный ресурс] // KPMS Менеджмент качества. URL: https://www.kpms.ru/Automatization/CASE_tools.htm (дата обращения 25.05.2021);
5. Введение в UML: Информация [Электронный ресурс] // Национальный Открытый Университет «ИНТУИТ». URL: https://www.intuit.ru/studies/courses/1007/229/info (дата обращения: 25.05.21);

DESIGN AND DOCUMENTATION OF THE INFORMATION SYSTEM FOR AN OFF-SITE CAR WASH

Sukhanov Ivan D.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, vanekk2000@gmail.com

Abstract. The main tools for designing and documenting an information system for a car wash are considered. The study analyzed: the Internet audience, the mobile app market for 2020, and similar apps. It was revealed that in 2020, the average daily Internet audience in Russia was 65% of the country's population [1]. These statistics show that people in Russia are beginning to use smartphones and tablets more than personal computers. This means that more mobile apps are being used. Statistics show that during the period of restrictive measures due to "Covid-19", the market of mobile applications that provide services of various kinds increased by 199% [2]. As a result of the analysis of analogs of information systems for off-site car washes, the main requirements for our information system were presented: a user-friendly interface, the ability to add multiple cars (according to statistics [3] for some people, this is an actual function), the function of automatically determining the location and building optimal routes for staff, a convenient form for filling out an application for a car wash, a system of discounts for customers and a system of motivation for staff.

As a result of the analysis of design and development tools, an instrumental software for designing an information system for an on-site car wash was selected: CASE-a StarUML tool for modeling the system [4], [5]; a web service Figma.ru to develop a prototype of the application interface for customers and staff. As a result of designing an information system for an on-site car wash, we described the behavior of the system and the user's interaction with it, both on the part of customers and on the part of staff. To describe the interactions of the client or staff with the system, we used: use case diagrams, activity diagrams, sequence diagrams, and described the order acceptance algorithm using a flowchart. Based on the diagrams we were able to design a possible system interface that contains the functionality described in the diagrams.

Keywords: tools for design and documentation, off-site car wash, user-friendly interface, mobile application design, tool software, CASE tool, customer application, staff application, use case diagram, sequence diagram, flowchart, interface design.

УДК 004.055

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ АВТОМАТИЗАЦИИ АЛГОРИТМОВ ДИАЛОГОВЫХ ИНТЕРФЕЙСОВ ДЛЯ ВНЕШНИХ СИСТЕМ

Летовальцев Даниил Дмитриевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, daniil-letovalcev@yandex.ru

Рассматриваются основные средства для проектирования и документирования информационной системы для автоматизации алгоритмов диалоговых интерфейсов для внешних систем. В ходе исследования были проанализированы: преимущества и недостатки аналогов, спрос на подобные информационные системы и экспериментально доказана востребованность на рынке систем для автоматизации алгоритмов диалоговых интерфейсов. При анализе аналогичных информационных систем были выявлены основные недостатки, которые требуется устранить в реализуемой информационной системе: слабая интеграция с ВКонтакте, отсутствие адаптивной мобильной версии, отдельная регистрация на электронном ресурсе, перегруженный интерфейс. В результате обзора инструментального ПО было выбрано инструментальное программное обеспечение информационной системы: CASE-средство PlantUML для моделирования системы [1]; сервис Figma для прототипированния интерфейса приложения; React.JS, HTML5, CSS3 для клиентской части информационной системы; Node.js как платформа для реализации серверной части информационной системы; MongoDB, Redis как СУБД на серверной части. В результате проектирования описываемой информационной системы было описано поведение системы и взаимодействие с ней пользователей. На основе построенных диаграмм был спроектирован интерфейс информационной системы, интерфейс был спроектирован как под десктопные, так и под мобильные устройства. На Android и iOS были соблюдены принципы дизайна в интерфейсе для этих операционных систем [2], [3]. Так же был проведен пилотный запуск информационной системы, во время которого за полтора месяца было достигнуто 52.000 уникальных пользователей и 8.629 уникальных пользователей в день максимум. Результаты пилотного запуска отображают востребованность проектируемой информационной системы на рынке систем для автоматизации алгоритмов диалоговых интерфейсов, демонстрируют спрос и актуальность системы.

Ключевые слова: средства для проектирования и документирования, конструктор чат-ботов, чат боты, автоматизация диалоговых интерфейсов, ВКонтакте, VKUI, адаптивный дизайн, диалоговые интерфейсы, автоматизация алгоритмов, MERN.

Введение

В эпоху социальных сетей, когда одним из крупнейших каналов связи между представителями услуг и потребителями являются те самые социальные сети, автоматизация диалоговых интерфейсов становится актуальна как никогда. Помимо представителей услуг в потребности автоматизации диалоговых интерфейсов например являются:

- Студенты и школьники (например: отображение расписание образовательного учреждения)
- Администраторы страниц в социальных сетях (например: отображение актуальной информации о ценах, времени работы)
- Представители сферы бизнеса (например: отображение справочной информации о предприятии)

Автоматизация диалоговых интерфейсов позволяет решать множество бизнес-задач начиная от отображения общей информации о бизнесе заканчивая приемом и оплаты услуг.

Анализ предметной области

Информационная система для автоматизации алгоритмов диалоговых интерфейсов для внешних систем включает в себя процесс, при котором пользователь ИС может создать чат-бота, а так же настроить и развернуть его. Определим основные процессы, которые пользователь должен выполнить при ручном создании чат-бота:

- Написание программного кода
- Тестирование и отладка программного кода
- Настройка Callback API ВКонтакте
- Создание токенов сообщества и их настройка
- Аренда сервера для развертывания
- Развертывание чат-бота
- Настройка сервера

Дополнительные процессы, которые возможно потребуется реализовать пользователю в зависимости от его требований:

- Хранение информации о пользователях чат-бота
- Проведение рассылки в чат-боте
- Создание панели администрирования чат-бота
- Управление доступом в панель администрирования
- Гибкая настройка чат-бота, через панель администрирования
- Логирование событий в чат-боте для дальнейшего анализа
- Формирование статистики чат-бота

Требования

Должна быть спроектирована информационная система для автоматизации алгоритмов диалоговых интерфейсов для внешних систем. Система будет предназначена для создания чат-ботов для сообществ социальной сети ВКонтакте, которую стоит использовать в силу её большой аудитории и доступности API для интеграции решений.

К требованиям системы можно отнести:

- Предоставления пользователям возможности создания, редактирования и управления чат-ботом
- Развертывание созданного пользователем чат-бота на сервере
- Обеспечение конфиденциальности, целостности и доступности информации
- Интеграция с ВКонтакте
- Адаптировать систему под мобильные устройства для использования с комфортом
- Разработать приятный и комфортный в использовании UI / UX

- Обеспечить возможность генерации отчетов с экспортом в актуальные форматы данных

- Ведение логов о событиях в информационной системе для дальнейшего анализа

- Использовать современные технологии в реализации, обеспечить гибкость разработки системы и её масштабируемость

Обзор инструментального ПО

В качестве языка программирования выберем JavaScript, одним из главных преимуществ это будет, то что имеется возможность использования платформы Node.js, которая даст нам выигрыш в скорости работы нашей ИС, больший чем например Java. Так же учитывая, то что наше клиентская составляющая будет так же на JavaScript, то одинаковый язык для клиентской и серверной составляющей является хорошим плюсом в нашем выборе. Node.js – платформа, позволяющая использовать JavaScript в серверных приложениях. Главный плюс Node.js в его быстроте и возможности масштабируемости, чем он например выигрывает перед Java. В Node.js нет модели параллелизма на основе потоков. Когда в Java на каждый запрос на веб-сервера создается поток, то при большом количестве запросов количество создаваемых потоков соответственно будет расти и процессор сервера при большом количестве пользователей будет занят только переключением потоков. В Node.js же не создается новый поток, в Node.js один поток, однако он будет выигрывать в скорости перед Java в рамках данной задачи. На сервер будет идти большое количество HTTP-запросов при запросе пользователей к ботам ИС, при реализации на Node.js не будет создаваться поток на каждого пользователя, что существенно ускорит работу нашего сервера и с точки зрения бизнеса снизит расходы на аппаратное обеспечение сервера. В Node.js реализована событийно-ориентированная архитектура, благодаря которой снижается потребление памяти, упрощается реализация серверной логики. С серверной стороны будет: Node.js из-за выигрыша в скорости, Redis в качестве локального хранилища ключ-значение, MongoDB в качестве основной СУБД. С клиентской стороны будет использован: React.js, HTML5, CSS3. Для проектирования и прототипирования интерфейса будет использован: Figma. Для создания диаграмм будет использован: PlantUML. В целом получается известный стек (набор технологий) MERN (MongoDB, Express.js, Reat.js, Node.js), который хорошо зарекомендовал себя и является популярным решением при создании веб-сервисов.

Моделирование информационной системы

В информационной системе будет реализовано клиент-серверное взаимодействие через API. Так как наша система будет интегрирована с ВКонтакте, более глубоко, чем аналоги, то регистрация пользователей для создания бота нам будет не нужна. Нужно будет только проверять подпись пользователя, о том что это действительно он послал запрос. На каждый прецедент нам нужен будет отдельный метод обработчик на сервере. При открытии страницы будет посылаться запрос на сервер для получения данных требуемых для отображения на странице приложения. На сервере будет проверяться подпись пользователя, после этого будет получен список данных пользователя из базы данных и будет возвращен ответ в клиентскую часть приложения. Клиентская часть в зависимости от ответа будет отображать: либо заглушку, о том что у пользователя ещё нет данных; либо отображать список данных.

Прототип пользовательского интерфейса

В проектируемом интерфейсе будут отображены элементы управления для пользователей информационной системы. Интерфейс должен быть простым и понятным для пользователей и в нем должны быть учтены привычные для пользователя паттерны поведения. С помощью интерфейса рядовой пользователь интуитивно и без затруднений должен понять как создать чат-бота для своего сообщества и разобраться в его управлении. Ведь в интерфейсе реализуемой информационной системы должно быть учтено поведение пользователей не имеющих технических навыков и быть понятной им, потому что проектируемая система рассчитана на массовую аудиторию и чтобы любой пользователь смог создать себе чат-бота, который бы покрывал его потребности.

Библиографический список

1. Введение в UML: Информация [Электронный ресурс] // Национальный Открытый Университет «ИНТУИТ». URL: https://www.intuit.ru/studies/courses/1007/229/info (дата обращения: 25.05.21);
2. Design for Android [Электронный ресурс] // Google Developers. URL: https://developer.android.com/design (дата обращения 25.05.2021);
3. Human Interface Guidelines [Электронный ресурс] // Apple Developers. URL: https://developer.apple.com/design/human-interface-guidelines/ (дата обращения 25.05.2021);

DESIGNING AND DOCUMENTING AN INFORMATION SYSTEM FOR AUTOMATING DIALOG INTERFACE ALGORITHMS FOR EXTERNAL SYSTEMS

Letovaltsev Daniil D.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, daniil-letovalcev@yandex.ru

Abstract. The main tools for designing and documenting an information system for automating algorithms of dialog interfaces for external systems been considered. The study analyzed the advantages and disadvantages of analogs, the demand for such information systems, and experimentally proved the market demand for systems for automating the algorithms of dialog interfaces. When analyzing other information systems, the main shortcomings that need to be eliminated in the implemented information system were identified: weak integration with VKontakte, the lack of an adaptive mobile version, separate registration on site, and an overloaded interface. As a result of the review of the tool software, the information system tool software was selected: PlantUML for system modeling; Figma for prototyping UI; React.JS, HTML5, CSS3 for the client side of the information system; Node.js as a platform for implementing the server part of the information system; MongoDB, Redis as a DBMS on the server part. As a result of the design of the described information system, the behavior of the system and the interaction of users with it were described. Based on the constructed diagrams, the interface of the information system was designed, the interface was designed for both desktop and mobile devices. On Android and iOS, the design principles in the interface for these operating systems were followed. There was also a pilot launch of the information system, during which 52,000 unique users were reached in a month and a half, and a maximum of 8.629 unique users per day. The results of the pilot launch reflect the demand for the designed information system in the market of systems for automating the algorithms of dialog interfaces, demonstrate the demand and relevance of the system.

Keywords: design and documentation tools, chatbot constructor, chatbots, chat bots, VKontakte, automating dialog interfaces, VKUI, adaptive design, dialog interfaces, MERN.

УДК 004.62

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ СИСТЕМЫ ДЛЯ СОЗДАНИЯ, ХРАНЕНИЯ И ВОССТАНОВЛЕНИЯ РЕЗЕРВНЫХ КОПИЙ БАЗ ДАННЫХ MYSQL

Тудвасев Илья Васильевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, tudvasevilya@gmail.com

В работе производится анализ предметной области, в результате которого определяется классификация резервного копирования. Также анализируются существующие системы, предназначенные для резервного копирования баз данных MySQL. На основе выделенных достоинств и недостатков существующих систем выдвигаются требования к собственной проектируемой системе. Производится выбор средств проектирования и разработки. В качестве средств проектирования рассматриваются CASE-средства для UML-моделирования. Перед выбором средств разработки сравниваются различные способы реализации. Далее рассматриваются готовые консольные решения для резервного копирования и восстановления баз данных. Сравниваются утилиты от MySQL Server. На следующем шаге производится проектирование системы. Выделяются типы пользователей. На основе ранее определенных требований и выделенных типов пользователей строится диаграмма прецедентов. Для объяснения некоторых прецедентов также строятся диаграммы последовательностей. Для описания процессов резервного копирования и восстановления базы данных строятся диаграммы последовательностей. Для графического представления структуры компонентов системы и взаимодействующих узлов строится диаграмма развертывания. На основе построенных моделей системы проектируется пользовательский интерфейс.

Ключевые слова: проектирование и документирование системы, резервное копирование баз данных, хранение резервных копий баз данных, восстановление баз данных, облачное хранилище, MySQL, серверный скрипт, UML-моделирование, проектирование интерфейса.

Введение

На данный момент многими практикуется работа с базами данных. Повреждение данных на носителе или же сбой на сервере может привести к потере данных. В качестве решения этой проблемы были придуманы различные способы создания резервных копий. Но также существует проблема хранения резервных копий. Хранить резервные копии на том же сервере или носителе является небезопасным решением, так как сбой может привести как к потере самой базы данных, так и к потере ее резервных копий. Для решения этих проблем производится проектирование и документирование собственной системы резервного копирования.

Анализ предметной области

В результате анализа резервного копирования баз данныхMySQL выделяется классификация процесса. Резервное копирование классифицируется:

- по типу создаваемой копии;
- по методу создания копии;
- по полноте сохраняемой информации.

По типу создаваемой копии выделяются физическое и логическое резервное копирование. По методу создания копии – холодное и горячее. По полноте сохраняемой информации – полное, инкрементное и дифференциальное.

На следующем этапе анализируются существующие системы, предназначенные для резервного копирования баз данных MySQL. Рассматриваются следующие системы: phpMyAdmin [2], Sypex Dumper [3], Adminer [4], Handy Backup [5], HeidiSQL [6]. В результате анализа выделяются основные достоинства и недостатки существующих систем. На основе выделенных достоинств и недостатков выдвигаются требования к собственной проектируемой системе:

— интуитивно понятный интерфейс;
— поддержка Windows и Linux ОС;
— экономия памяти;
— возможность переключения между языками (русский/английский);
— возможность просмотра журнальных записей;
— предоставление функций создания резервной копии, восстановления БД из резервной копии;
— возможность выбора объектов для резервного копирования;
— возможность контролировать процесс (пауза/полная остановка);
— возможность добавления облачного сервера для хранения копий;
— использование технологии, позволяющей делать быстрые бэкапы;
— возможность редактирования программной составляющей.

Выбор средств проектирования и разработки

После анализа предметной области производится выбор средств проектирования и разработки. В качестве средств проектирования рассматриваются CASE-средства для UML-моделирования [7]: StarUML, Draw.IO, Umbrello, Microsoft Visio. Средства сравниваются по удобству интерфейса, функционалу, поддержке русского языка и способу распространения. В результате сравнения выбирается CASE-средство Draw.IO. Передвыбором средств разработки сравниваются различные способы реализации, среди которых выбирается реализация в виде серверного скрипта. Для серверной части в качестве языка программирования выбирается PHP. Для клиентской части – язык гипертекстовой разметки HTML5 и фреймворк Bootstrap.

Далее рассматриваются готовые консольные решения для резервного копирования и восстановления баз данных. Сравниваются утилиты от MySQL Server [8]: mysqldump, mysqlpump, mydumper, MySQL Shell Dump/Load. В результате сравнения выбирается утилита MySQL Shell Dump/Load, так как она имеет наиболее высокую скорость работы, а также поддерживает больше функций, подходящих под ранее установленные требования к собственной системе.

Проектирование информационной системы

Проектирование информационной системы состоит из следующих этапов:

— построение модели системы;
— проектирование пользовательского интерфейса системы.

Перед построением диаграммы прецедентов определяются действующие лица. Для действующих лиц производится распределение прецедентов. Также выделяются вложенные прецеденты. В результате строится следующая диаграмма:

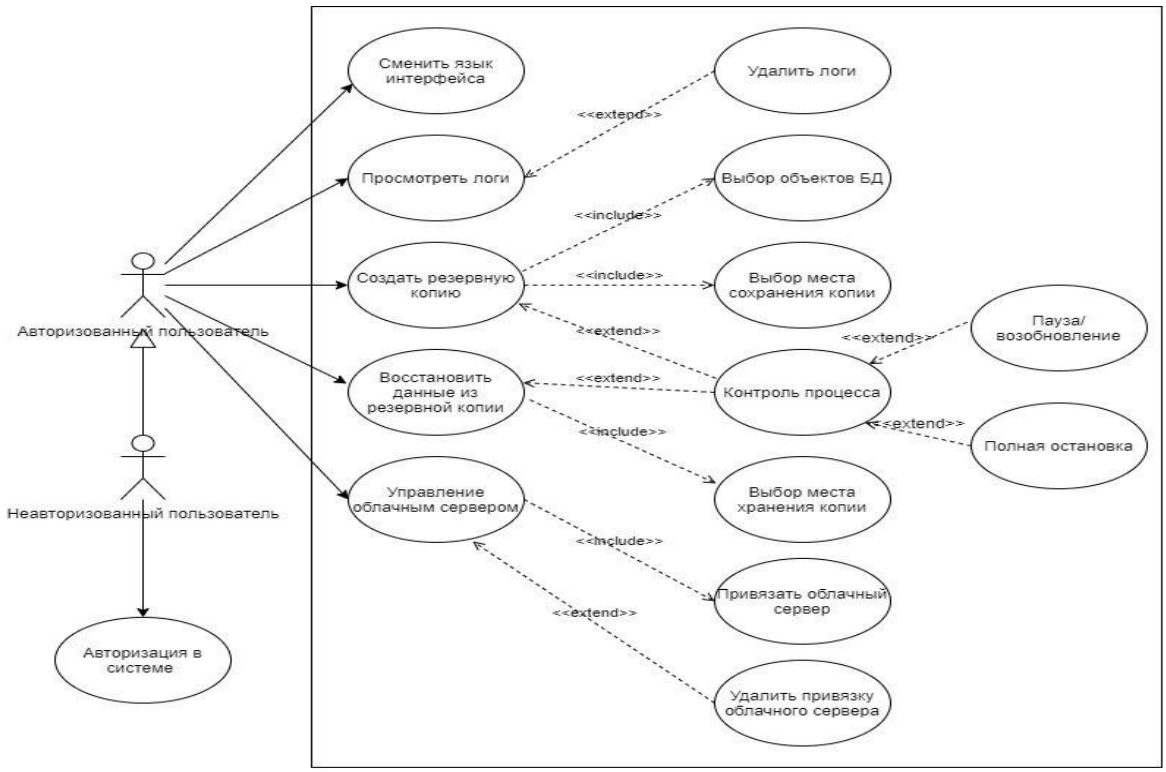


Рис. 1 – Диаграмма прецедентов

Для объяснения прецедентов «Создать резервную копию», «Восстановить данные из резервной копии» и «Привязать облачный сервер» строятся диаграммы последовательностей. Для описания процессов создания резервной копии и восстановления БД из резервной копии строятся диаграммы взаимодействия. Для графического представления структуры компонентов системы и взаимодействующих узлов строится диаграмма развертывания.

На основе построенных моделей системы проектируется пользовательский интерфейс с помощью ранее выбранных средств: HTML5 и Bootstrap (см. рис. 2).

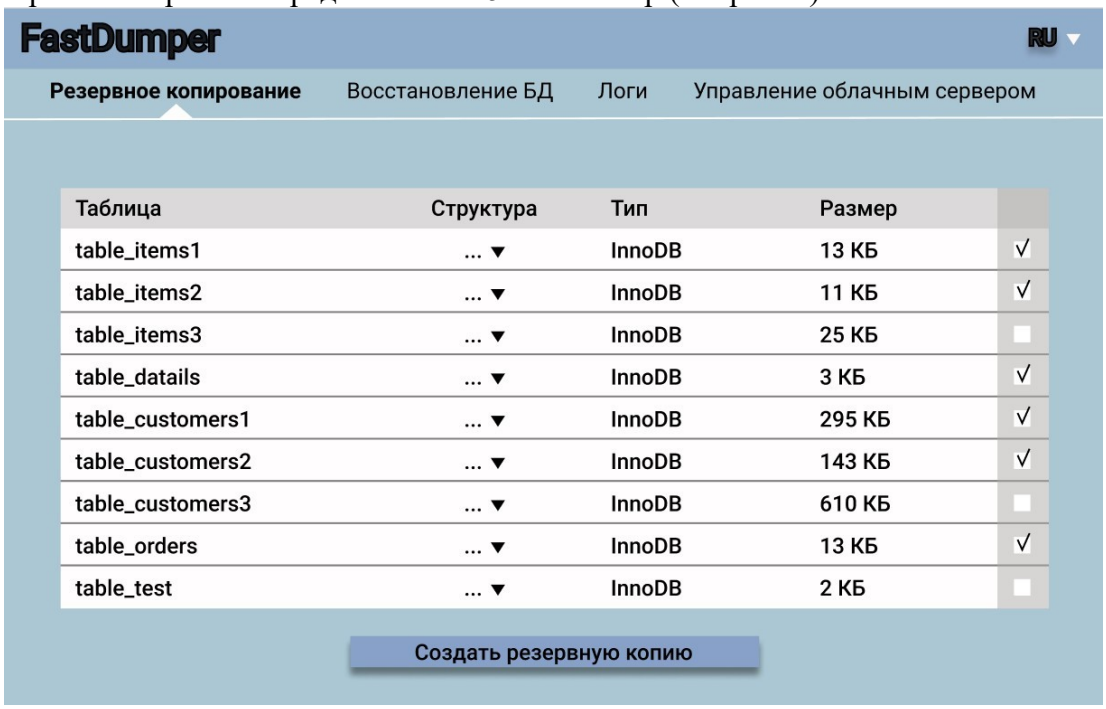


Рис. 2 – Прототип пользовательского интерфейса

Заключение

В результате выполнения работы выполнены следующие задачи:

— анализ существующих аналогов ИС резервного копирования БД MySQL;
— обзор инструментального ПО;
— проектирование и документирование ИС резервного копирования БД MySQL;
— разработка прототипа пользовательского интерфейса.

Разработанная модель информационной системы позволяет за короткое время создавать резервные копии, восстанавливать БД MySQL из резервных копий, а также сохранять копии на облачном, серверном или локальном хранилище. Также система отображает журнальные записи и позволяет пользователю управлять ими.

Библиографический список

1. Глава 8. Резервирование и восстановление // RLDP – русская версия Linux Documentation Project, включающая в себя руководства по СУБД MySQL [Электронный ресурс]. URL: http://www.rldp.ru/mysql/mysql80/backup.htm (Дата обращения: 16.11.20)
2. phpMyAdmin Приложение для администрирования СУБД MySQL [Электронный ресурс]. URL: https://www.phpmyadmin.net/ (Дата обращения: 03.12.20)
3. Sypex Dumper Программный продукт для создания и восстановления резервных копий баз данных MySQL [Электронный ресурс]. URL: https://sypex.net/ (Дата обращения: 03.12.20)
4. Adminer – database management in a single PHP file [Электронный ресурс]. URL: https://www.adminer.org/ (Дата обращения: 03.12.20)
5. Handy Backup Программа для резервного копирования и восстановления данных ПК [Электронный ресурс]. URL: https://www.handybackup.ru/ (Дата обращения: 05.12.20)
6. HeidiSQL Free and open-source administration tool for MySQL and its forks [Электронный ресурс]. URL: https://www.heidisql.com/ (Дата обращения: 05.12.20)
7. Введение в UML: Информация // Национальный Открытый Университет «ИНТУИТ» [Электронный ресурс]. URL: https://www.intuit.ru/studies/courses/1007/229/info (Дата обращения: 20.05.21)
8. MySQL Shell Dump & Load // MySQL Server Blog – news from MySQL Server Team [Электронный ресурс]. URL: https://mysqlserverteam.com/mysql-shell-dump-load-part-1-demo/ (Дата обращения: 20.05.21)

DESIGN AND DOCUMENTING A SYSTEM FOR CREATING, STORING AND RESTORING BACKUPS OF MYSQL DATABASES

Tudvasev Ilya V.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, tudvasevilya@gmail.com

The paper analyzes the subject area, as a result of which the classification of backup is determined. The existing systems designed for backup of MySQL databases are also analyzed. Based on the identified advantages and disadvantages of existing systems, the requirements for the own designed system are put forward. The choice of design and development tools is made. CASE-tools for UML modeling are considered as design tools. Before choosing the development tools, different implementation methods are compared. Next, we consider ready-made console solutions for database backup and recovery. Utilities from MySQL Server are compared. The next step is the design of the system. The types of users are highlighted. A use case diagram is constructed based on previously defined requirements and selected user types. Sequence diagrams are also constructed to explain some use cases. Sequence diagrams are constructed to describe the processes of database backup and recovery. A deployment diagram is constructed to graphically represent the structure of the system components and interacting nodes. Based on the constructed models of the system, the user interface is designed.

Keywords: system design and documentation, database backup, database backup storage, database recovery, cloud storage, MySQL, server script, UML modeling, interface design.

УДК 004.65

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ РАБОТЫ ОТДЕЛА КАДРОВ

Лискова Екатерина Сергеевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, liskova98@mail.ru

Аннотация. Лискова Екатерина Сергеевна, направление «Фундаментальные информатика и информационные технологии», группа ММ/О – 1,2 – 2018 НБ, «Проектирование и документирование информационной системы работы отдела кадров», 2021 год, кафедра Информационных технологий ПГНИУ.

В работе описан процесс проектирования системы учета данных персонала, работающего (или планирующего работать) в компании. Данная система будет необходима для любой компании, целью которой является возможность удобного учета данных персонала компании, его подбор, а также взаимодействие с ним. Исследованы существующие информационные системы, выявлены их плюсы и минусы. Спроектирована собственная ИС для удобной подачи документов (резюме, заявления) кандидатами. В разработке использовался опыт работы с базами данных, CASE-средствами и инструментами WEB-дизайна.

Ключевые слова: отдел кадров, система учета данных, информационная система.

Введение

Актуальность данной работы заключается в том, что до сих пор взаимодействие отдела кадров и непосредственно кандидатов/специалистов зачастую происходит очно, когда этого можно было бы избежать.

Целью данной работы является спроектированная информационная система учета данных сотрудников и стажерского состава, работающих или желающих работать в компании, а также позволяющая удобно взаимодействовать с документами сотрудников, как уже работающих специалистов, так и новых кандидатов.

Для постановки задач на проектирование системы, необходимо рассмотреть уже существующие ИС учета данных сотрудников, выявить их преимущества и недостатки. На сегодняшний день существует множество систем, которые предлагают решения для автоматизации работы кадровых отделов. Рассмотрим основные типы этих систем на примере нескольких продуктов из этой области.

Для более наглядного представления о различиях рассмотренных систем, ниже представлена сравнительная таблица, где плюсом отмечено соответствие системы указанному требованию.

Таблица 1 – Сравнение существующих ИС

Название ИС	ИС «БОСС-КАДРОВИК»	ПМ «1С: Зарплата и Кадры»	КИС «Флагман»
Базовый функционал	+	+	+
Неперегруженность	+	-	-

Интуитивная понятность	-	+	-
Дистанционность	-	-	+
Пробная версия	-	-	+

Можно сделать вывод, что на данный момент «идеальной» системы не существует и есть большой потенциал роста у каждой из систем. Также это говорит о том, что таким системам можно и нужно составлять конкуренцию на рынке.

Выбор CASE-средства

Для проектирования ИС необходимо выбрать средства проектирования статических и динамических аспектов системы, а также средства для проектирования базы данных и интерфейса.

Проектирование системы можно осуществлять при помощи разновидностей UML (Unified Modeling Language) диаграмм. UML – унифицированный язык моделирования. Применяется для использования визуализации, спецификации, проектирования и документирования программных систем.

Из существующих систем для работы с UML были выбраны для сравнения пять наиболее популярных решений.

Из сравнительной таблицы видно, что выбранным критериям соответствуют программы VisualParadigmExpressEdition и draw.io. Между двумя вариантами, было выбрано CASE-средство draw.io, потому как оно является более доступным и не требует регистрации.

Более наглядное сравнение представлено в таблице 2.

Таблица 2 – Сравнение CASE-средств проектирования ИС

	Creately	Lucidchart	VisualParadigm		draw.io
			ExpressEdition	Diagrams	
Необходимый функционал	+	+	+	+	+
Простота использования	+	-	+	+	+
Режим онлайн	+	+	+	+	+
Бесплатная версия	-	-	+	-	+
Экспорт диаграмм	-	+	+	-	+

Из сравнительной таблицы видно, что выбранным критериям соответствуют программы VisualParadigmExpressEdition и draw.io. Между двумя вариантами, было выбрано CASE-средство draw.io, потому как оно является более доступным и не требует регистрации.

Выбор средства для проектирования схемы БД

Для удобного функционирования ИС учета данных кандидатов необходимо создание базы данных (БД), в которой будут содержаться личные данные кандидатов, их заявления на направления обучения, статусы обработки заявлений и т.д. Для реализации будет необходимо построить схему взаимосвязей и нормализовать БД, то есть привести к третьей нормальной форме.

Для построения схемы было выбрано приложение dbdiagram.io, которое позволяет выгружать схему БД как в форматах .pdf и .png, так и в виде кода для PostgresSQL, MySQL и SQL Server. Еще одним преимуществом данной среды проектирования является наличие простого и удобного функционала для разработки.

Выбор средств разработки

К выбору средств разработки стоит подойти более основательно. Если выбор средств проектирования ограничивается только созданием статичных схем и диаграмм, то разработка включает в себя тестирование, отладку, поддержку системы в рабочем состоянии, возможность перестраивания и расширения функционала системы.

Важным этапом проектирования ИС является создание БД. Для учета данных кандидатов, подающих резюме в компанию, это является основной частью проектирования,

потому как необходимо хранить и обрабатывать большой объем данных. Поэтому от выбора СУБД будет зависеть основная часть работы системы.

На рассмотрение вынесены три самые популярные СУБД: SQLite, MySQL и PostgreSQL.

SQLite – компактная встраиваемая система, которая отлично подходит для разработки и тестирования.

MySQL – это самая популярная из всех крупных серверных БД. Пользователи отмечают простоту в освоении и широкий функционал.

PostgreSQL – свободно распространяемая объектно-реляционная СУБД, ориентирующаяся в первую очередь на полное соответствие стандартам и расширяемость.

Из рассмотренных СУБД предлагается использовать PostgreSQL. Исходя из анализа и сравнения, система показывает себя как наиболее надежное средство разработки. Данная база данных применима к реализации приложений, где в приоритете стоит надежность и целостность данных. И благодаря расширяемости, справляется выполнением сложных процедур.

Выбор языка программирования

В качестве средств разработки предлагается использовать языки HTML, CSS и NodeJS для разметки, ввода и вывода информации.

Выбор среды разработки

В качестве среды для разработки рассмотрим два варианта: SublimeText и VisualStudioCode.

SublimeText – текстовый редактор, предназначенный для создания и изменения текстовых данных в общем и текстовых файлов в частности. После ознакомительного периода необходимо приобрести лицензию. Однако версия SublimeText 3, выпущенная в 2017 году, является доступной для всех разработчиков. Основная особенность данного текстового редактора заключается в том, что в нем нет лишнего функционала, и в то же время в нем есть все, что может потребоваться разработчику. Если какой-то функционал отсутствует, то его всегда можно дополнить одним из множества бесплатных плагинов.

VisualStudioCode – редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности подстраивания под каждого пользователя: выбор темы интерфейса, настраивание сочетания клавиш для более быстрого доступа к функционалу, кастомизация файлов конфигурации. Продукт распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом.

Поскольку особой разницы между этими двумя интегрированными средами разработки не выявлено, то окончательный выбор остается в пользу VisualStudioCode, так как данный редактор является наиболее привычной средой для разработки.

Вывод

Подводя итоги анализа, проведенного во второй главе, стоит вынести отдельно полученные результаты. Набор средств, с помощью которых будет реализована ИС учета данных кандидатов, подающих свои резюме в компанию, включает в себя следующие инструменты:

draw.io – для построения UML диаграмм;

dbdiagram.io – для построения схемы БД;

PostgreSQL – в качестве СУБД;

HTML и CSS – клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса;

Nodejs – программно-аппаратная часть сервиса;

VisualStudioCode – среда для разработки.

Заключение

С помощью выбранных средств, были построены UML-диаграммы прецедентов, классов, состояний и последовательности, спроектирована схема базы данных и создан прототип пользовательского интерфейса.

В процессе выполнения работы были выполнены следующие задачи:

1. Исследование предметной области и анализ существующих ИС учета данных сотрудников компании;
2. Выбор инструментария для проектирования ИС;
3. Проектирование ИС.

Вместе с этим достигнута цель работы – спроектирована ИС учета данных кандидатов, подающих свои резюме на рассмотрение в компанию.

ИС предназначена также для снижения нагрузки на ответственных администраторов кадровой службы и на технических работников.

Работа может послужить основой для реализации проектирования системы учета данных кандидатов, а также для дальнейшего внедрения в эксплуатацию в компании с большим количеством сотрудников. Помимо этого, система предусматривает расширение и усовершенствование своего основного функционала.

Библиографический список

1. Федеральный закон от 19.04.1991 № 1032-1-ФЗ «О занятости населения в Российской Федерации». Доступ из справ.-правовой системы КонсультантПлюс.
2. URL: http://www.consultant.ru/document/cons_doc_LAW_60/
3. Бабич А. Введение в UML [Электронный ресурс]. Режим доступа: http://www.intuit.ru/studies/courses/1007/229/info (дата последнего обращения: 03.04.2021)
4. Фаулер М. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. С. 27-29
5. SQLite, MySQL и PostgreSQL: сравнение популярных реляционных СУБД URL: https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/ (дата последнего обращения: 15.04.2021).
6. The Positive and Negative Aspects of Node.js Web App Development URL: https://www.mindinventory.com/blog/pros-and-cons-of-node-js-web-app-development/ (дата последнего обращения: 24.04.2021)
7. О продукте «Босс-Кадровик» URL: https://www.boss.ru/products/bk-about/ (дата последнего обращения: 20.04.2021)
8. «1С: Зарплата и управление персоналом 8». О продукте. URL: https://v8.1c.ru/hrm/ (дата последнего обращения: 26.04.2021)
9. «КИС "ФЛАГМАН"». О продукте. URL: http://infosoft.ru/o-kompanii

DESIGN AND DOCUMENTATION OF THE INFORMATION SYSTEM FOR THE WORK OF THE HR DEPARTMENT

Liskova Ekaterina S.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, ivanov@email.ru

Abstract. Liskova Ekaterina Sergeevna, direction "Fundamental informatics and information technologies", group MM / O – 1,2 – 2018 NB, "Design and documentation of the information system of the work of the personnel department", 2021, Department of Information Technologies, Perm State National Research University.

The paper describes the process of designing a data recording system for personnel working (or planning to work) in the company. This system will be necessary for any company, the purpose of which is the ability to conveniently record the data of the company's personnel, their selection, as

well as interaction with them. The existing information systems are investigated, their pros and cons are revealed. We have designed our own IS for convenient submission of documents (resumes, applications) by candidates. The development used experience with databases, CASE-tools and WEB-design tools.
Key words: personnel department, data accounting system, information system.

УДК 004.415

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТРИОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ УЧЁТА ДОГОВОРОВ И КОНТРОЛЯ ЗА ИХ ИСПОЛНЕНИЕМ

Вахрушев Геннадий Сергеевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, gena.vaxrushev@gmail.com

В статье рассмотрена система, предназначенная для ведения реестра договоров, с возможностью помечать этапы, на которых может находиться договор. Сделан анализ некоторых информационных систем, выполняющих похожие функции. Определены функциональные возможности системы. Построены диаграммы: прецедентов, классов, состояний, деятельности для заключения договора, деятельности для работы с реестром, создан прототип интерфейса системы.

Ключевые слова: договор, реестр договоров, этап договора, пользователь, автоматизировать, модель системы.

Договоры – неотъемлемая часть человечества. Благодаря им мы фиксируем наши обязательства. В основном фиксация проходила на бумаге, что является неудобным, так как бумагу можно легко потерять, испортить, её нужно физически передавать, к тому же для её изготовления нужны деревья. Сейчас широко развиты информационные технологии и нет необходимости вести договоры на бумаге. Их можно вести на компьютере, что намного удобнее и практичнее. Также, вместе с учётом договоров, на том же компьютере очень удобно вести контроль за их исполнением. Поэтому проектирование системы, в которой можно вести учёт договоров и контроль за их исполнением – очень актуальная задача. Но всякая система должна быть задокументирована, что тоже необходимо сделать.

Объект – проектирование и документирование систем.

Предмет – информационная система учёта договоров и контроля за их исполнением.

Цель работы – проект информационной системы учёта договоров и контроля за их исполнением.

В области документирования информационных рассматривались книги Шикиной В.Е. [1] и Антонова О.Б [2]. Сравнение этих книг представлено в таблице 1.

Таблица 1 – Сравнение книг по документированию ИС

Критерий	Книга 1	Книга 2
Наличие стандартов	+	+
Требования к документации	+	-
Назначение документации	+	-
Описание документов	+	+
Краткость	+	+
Ясность	+	+

Из таблицы видно, что обе книги имеют в себе информацию о различных стандартах документации и описание документов, которые могут присутствовать в документации, к тому же всё описано кратко и ясно. Однако книга 1 содержит в себе информацию о назначении документации и требования к ней, также информация в ней более полная и подробная, поэтому при документировании используется эта книга.

После изучения книги выбран ГОСТ 34.602-89 и определены требования к документации:

- описание возможностей системы;
- фиксация принятых проектных решений;
- определение условий функционирования системы;
- предоставление информации об эксплуатации системы;
- регламентирование процедуры защиты информации.

По части проектирования будет использоваться анализ аналогов и совершенствование их с учётом их недостатков, так как имеет смысл не разработка системы с нуля, а использование то, что уже существует с последующим его совершенствованием

При проектировании информационной системы необходимо определить функции системы, процессы, автоматизируемы системой, а также место пользователя в системе.

Система должна обеспечивать:

- хранение договоров и связанных документов, что включает в себя работу с базой данных, в рамках которой автоматизируются добавление, изменение и удаление документов, а поиск документов, экспорт реестра документов и сортировка документов по какому-либо параметру;
- контроль времени исполнения договора, то есть система должна уведомлять пользователя, когда срок исполнения этапа, на котором документ находится в данный момент, подходит к концу.

Пользователь в системе должен:

- вводить данные документов;
- помечать этапы исполнения;
- отправлять договор на согласование.

В качестве аналогов разрабатываемой выбраны две системы: Citeck и Directum. Сравнение данных систем представлено ниже в таблице 2.

Таблица 2 – сравнение аналогов ИС

Критерий	Citeck	Directum
Единый реестр договоров и связанных документов, с возможность фильтрации и поиска	+	+
Контроль за исполнением договоров	+	+
Экспорт реестра в различных форматах	+	-
Согласование договоров	+	+
Генерация счетов и накладных	+	-
Создание документов по шаблонам	+	+
Создание поручения по договору	+	-
Критерий	Citeck	Directum
Бизнес требования	+	-
Служба сканирования и заполнения договоров	-	+

Из таблицы видно, что система Citeck имеет больший функционал по сравнению с системой Directum, к тому же она имеет большое количество других функций, не описанных в таблице, что усложняет её освоение.

Несмотря на то, что система Directum уступает системе Citeck, меньшее количество функций обеспечивает более быстрое и простое освоение системы, к тому же у неё есть полезная служба сканирования и заполнения договоров.

Разрабатываемая система должна сочетать некоторые функции системы Citeck и простоту системы Directum.

Логика работы с информационной системой показана на диаграмме прецедентов (рис. 1).

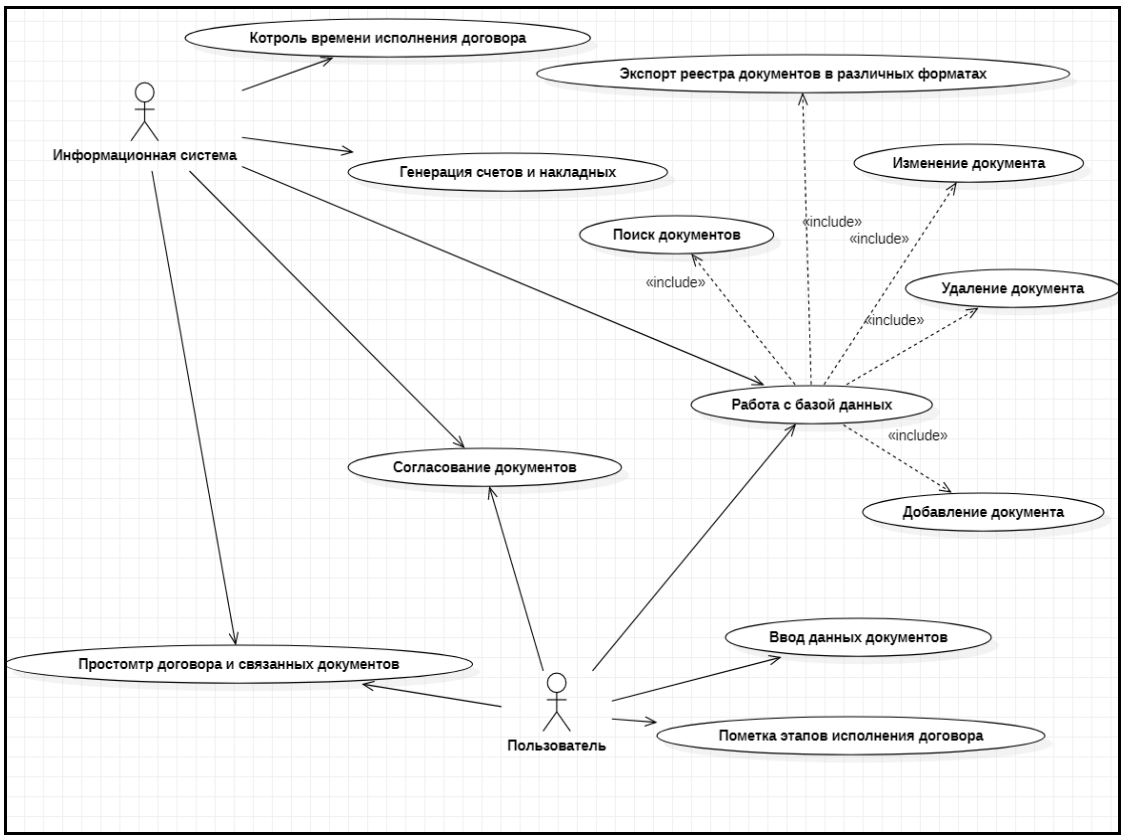


Рис 1. Диаграмма прецедентов

Структура классов системы, их атрибутов и взаимодействий показана на диаграмме классов (рис. 2).

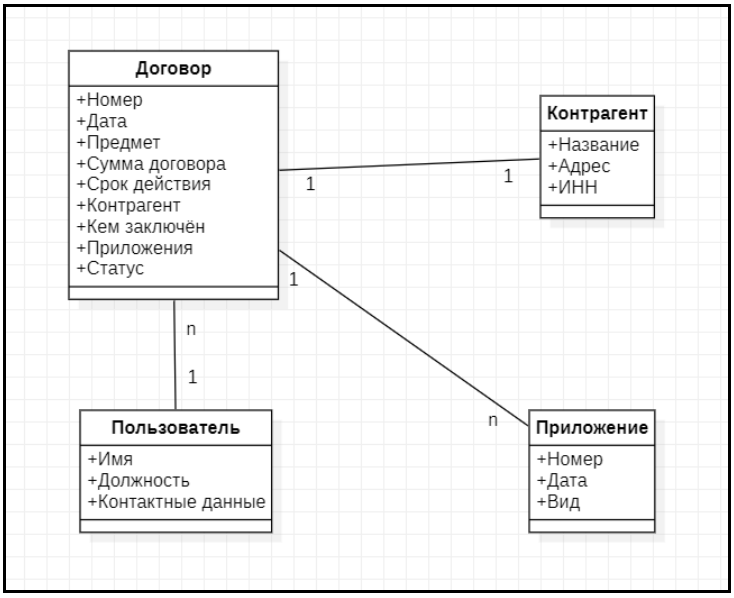


Рис. 2. Диаграмма классов

Из диаграммы классов видно, что в системе присутствуют 4 сущности:

1. Договор – содержит в себе информацию о договорах, из этих сущностей создаётся общий реестр договоров;
2. Контрагент – содержит в себе информацию о контрагентах;
3. Пользователь – содержит в себе информацию о работниках компании, заключающих контракты и использующих информационную систему;
4. Приложение – содержит в себе информацию о приложениях к каждому документу.

Договор может находиться на разных этапах, переходя от одного к другому. Ниже показана диаграмма состояний (рис. 3) для перехода договора от одного состояния к другому.

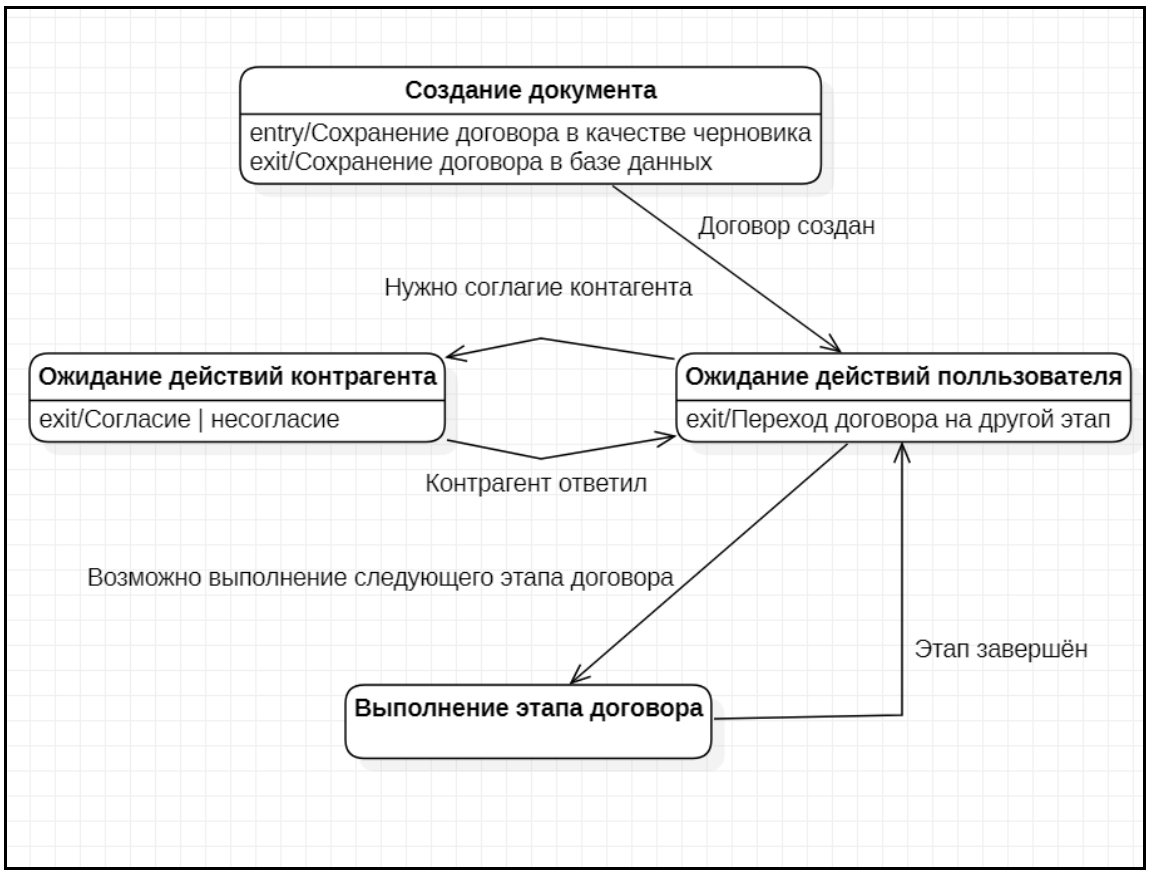


Рис. 3. Диаграмма состояний

Как видно из диаграммы сначала договор создаётся и, если договор не был создан до конца, он сохраняется в качестве черновика для последующего создания, в итоге, когда договор создан он сохраняется в базу данных. После создания договора он ожидает действий пользователя: либо отправляется к контрагенту для выражения его согласия или не согласия, например, при подписании договора или при завершении работ, либо переходит на следующий этап. После выполнения действий пользователя договор переходит на другой этап: либо на следующий, либо на предыдущий, если контрагент выразил своё несогласие.

На представленной ниже диаграмме деятельности (рис. 4) более подробно показан процесс взаимодействия пользователя и контрагента через информационную систему при заключении договора.

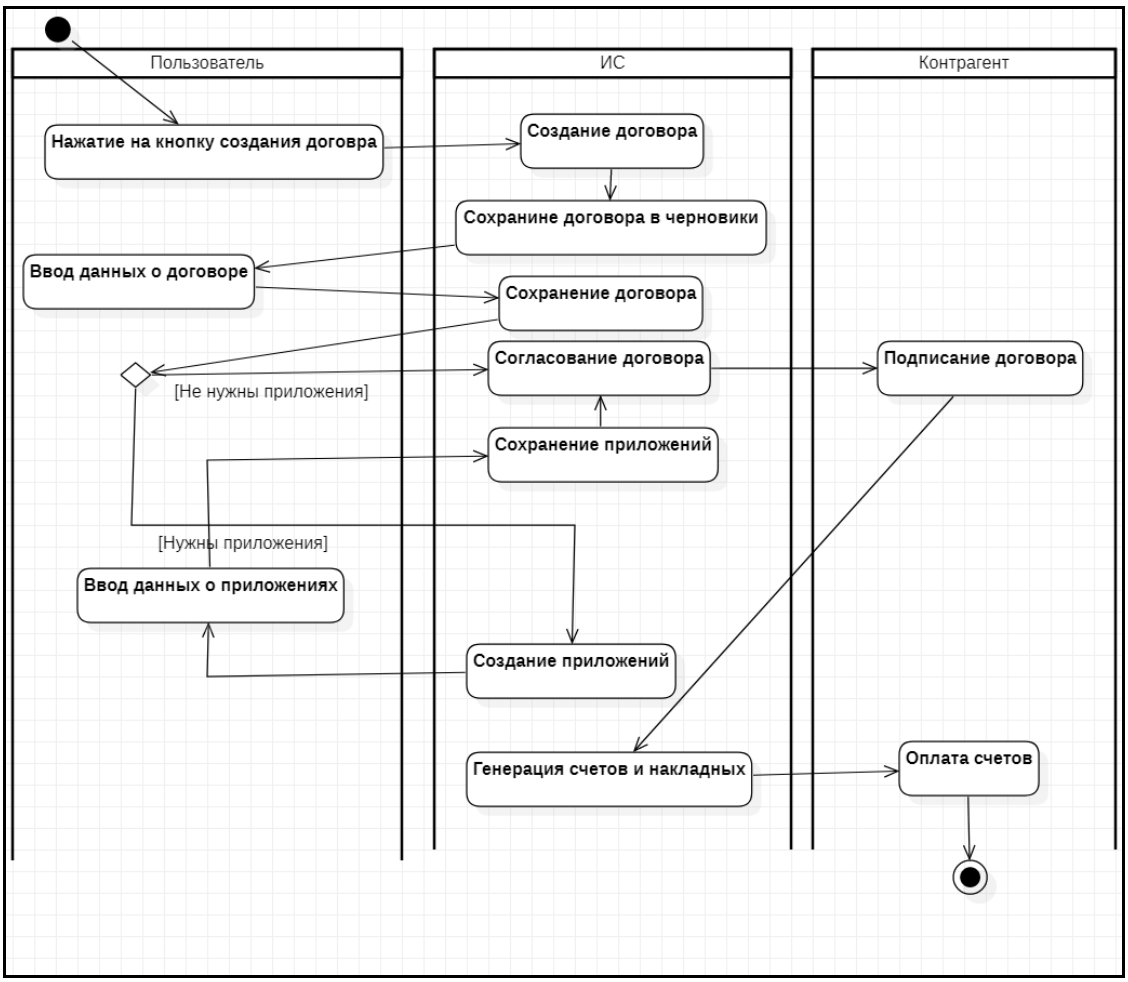


Рис. 4. Диаграмма деятельности для заключения договора

На данной диаграмме показано, что пользователь только вводит данные, контрагент только получает всё необходимое для подписания и оплаты, а информационная система создаёт и сохраняет всё необходимое.

На представленной ниже диаграмме деятельности (рис. 5) более подробна показана работа пользователя с реестром документов при помощи информационной системы.

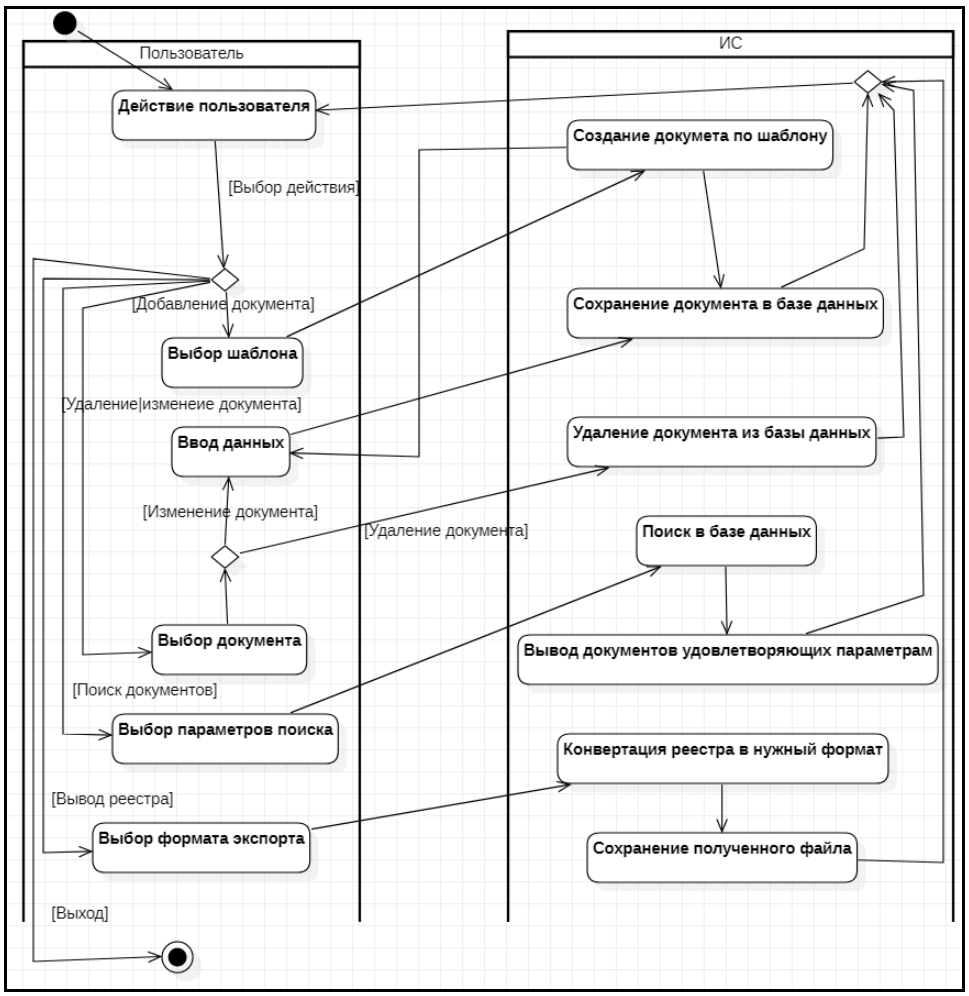


Рис.5. Диаграмма деятельности для работы с реестром

На диаграмме показано, что конкретно должен делать пользователь, чтобы выполнить то или иное действие с реестром документов, также показано что система должна делать с базой данных.

Ниже представлена схема базы данных информационной системы (рис. 6).

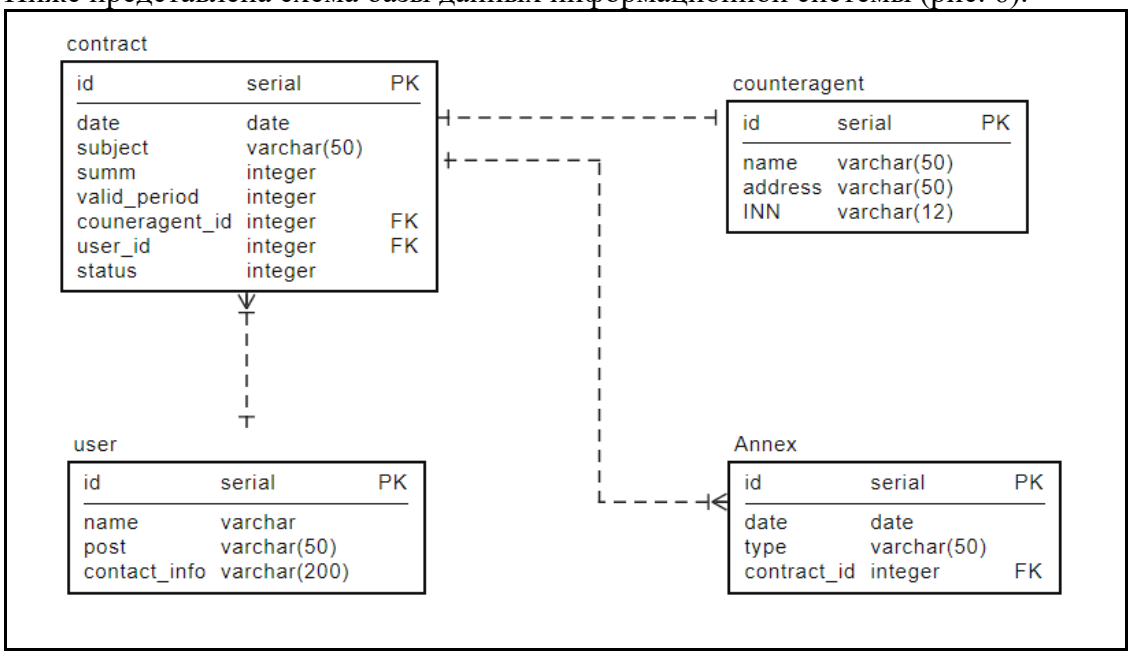


Рис. 6. Схема базы данных

Ниже представлены основные страницы дизайна приложения системы.

Рис.7. Окно регистрации

Рис. 8. Главная страница

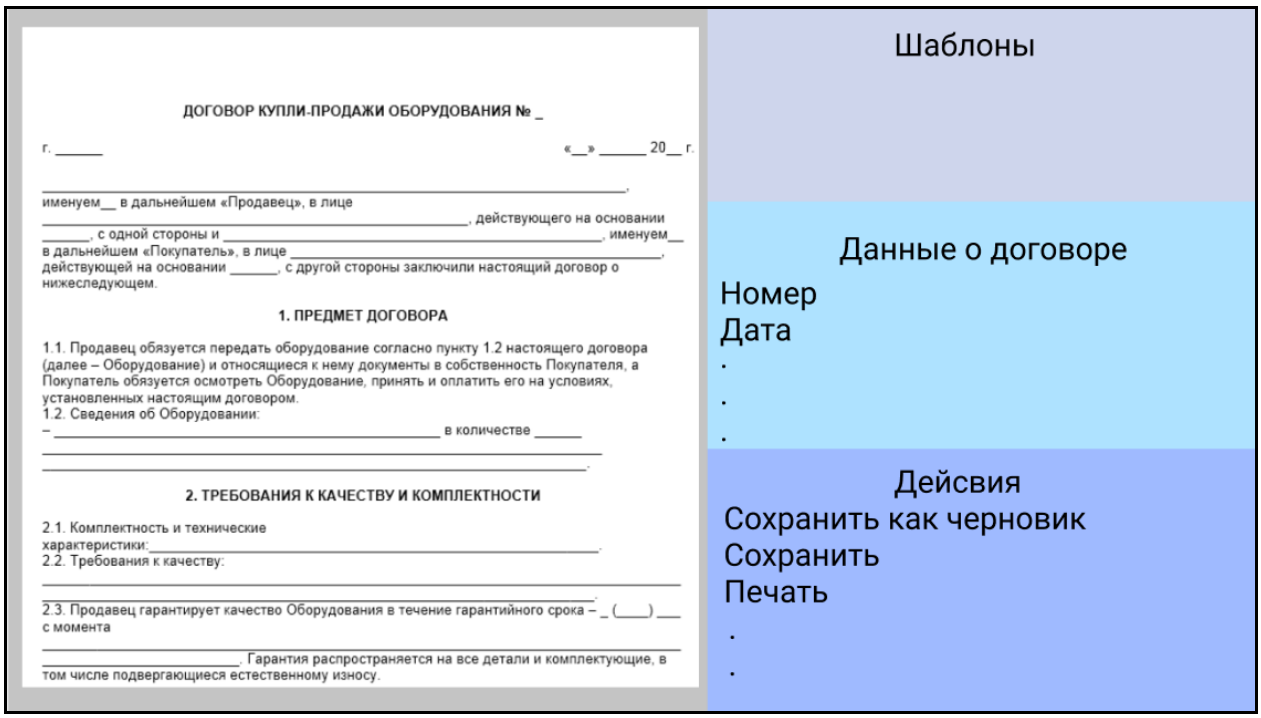


Рис. 9. Окно создания документа

Так как разрабатывается система для типовой структуры, а не для какой-то определённой организации, при документировании будут опущены некоторые пункты, например, плановые сроки начала и окончания работы по созданию системы, сведения об источниках и порядке финансирования работ и т.п., также по этой причине пропущен раздел «Общие сведения». В документации выделены: техническое задание и руководство пользователя.

В техническом задании выделены следующие пункты:

1. Назначение и цели создания (развития) системы
2. Характеристика объектов автоматизации
3. Требования к системе
    a. Требования к системе в целом
    b. Требования к структуре системы
    c. Требования к режимам функционирования системы
    d. Требования к надёжности
    e. Требования к эргономике и технической эстетике
    f. Требования к защите информации от несанкционированного доступа

В руководстве пользователя выделены следующие пункты:

1. Введение
    a. Область применения
    b. Краткое описание возможностей
    c. Уровень подготовки пользователя
2. Назначение и условия применения
3. Подготовка к работе
    a. Состав дистрибутива
    b. Запуск системы
    c. Проверка работоспособности системы
4. Описание операций
5. Аварийные ситуации
6. Рекомендации по освоению

Остальные пункты документации должны быть описаны в зависимости от конкретной организации.

Разработанная система предназначена для компаний, желающих упростить работу с договорами и перевести её в электронный формат.

Библиографический список

1. Шикина, Виктория Евгеньевна Техническая документация информационных систем : учебное пособие / В.Е. Шикина. – Ульяновск : УлГТУ, 2018. – 92 с
2. Антонов О. Б. Документирование информационных систем / О. Б. Антонов – «ЛитРес: Самиздат», 2019
3. Система управления договорами Citeck [электронный ресурс]
4. URL: https://www.citeck.ru/solutions/citeck-ecos-contract-management/ (Дата обращения: 28.05.2021)
5. Directum, управление договорами [электронный ресурс]
6. URL: https://www.directum.ru/solution/contracts_management (Дата обращения: 28.05.2021)

DESIGN AND DOCUMENTATION OF AN INFORMATION SYSTEM FOR ACCOUNTING FOR CONTRACTS AND MONITORING THEIR EXECUTION

Vakhrushev Gennadiy S.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, gena.vaxrushev@gmail.com

The article considers a system designed to maintain a register of contracts, with the ability to mark the stages at which the contract may be. The analysis of some information systems performing similar functions. The functional capabilities of the system are defined. Diagrams are constructed: use cases, classes, states, activities for concluding a contract, activities for working with the registry, a prototype of the system interface is created.

Keywords: contract, contract register, contract stage, user, automate, system model.

УДК 004.5

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИННОЙ СИСТЕМЫ ДЛЯ СОСТАВЛЕНИЯ МАРШРУТА В СЕТИ ОБЩЕСТВЕННОГО ТРАНСПОРТА Г. ПЕРМИ С СИСТЕМОЙ ПЕРЕСАДОК

Благиных Никита Владиславович

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, salenctawer@gmail.com

В данной статье рассмотрена система, которая помогает жителям города Перми составить маршрут из начальной точки в конечную в сети общественного транспорта. Обоснована актуальность проектирования автоматизированной системы для составления маршрута в сети общественного транспорта с системой пересадок. Сделан анализ похожих информационных систем. Определены функциональные возможности системы. Сделан выбор системы управления базой данных. Построены диаграмма вариантов использования, диаграмма действий, создан прототип интерфейса системы.

Ключевые слова: маршрут, сеть общественного транспорта, подбор транспорта, построение маршрута, автоматизировать, модель системы, система пересадок.

Сеть общественного транспорта города Перми стала более удобной для пассажиров за счет создания новых маршрутов, включения в маршрутную сеть новых улиц и остановок, а также продления некоторых маршрутов [1].

В условиях динамичного социально-экономического развития города, далеко не каждый житель может самостоятельно составить оптимальный маршрут передвижения по городу. Также немаловажную роль играет тот фактор, что добавляются новые улицы, остановки и сами транспортные маршруты меняют свои пути следования. Поиск всей новой информации может занять достаточно много времени [2].

В связи с этим, возникает необходимость в создании удобного, гибкого и простого приложения по поиску оптимального маршрута в сети общественного транспорта г. Перми с системой пересадок.

Актуальность данной темы состоит в том, что уже существуют информационные системы, выполняющие функцию составления оптимального маршрута, но они не всегда могут удовлетворять потребностям пользователей. Например, жители, которые плохо ознакомлены с новыми технологиями, продолжают самостоятельно искать информацию и передвигаться по не оптимальным маршрутам.

Цель данной работы состоит в проектировании и документировании информационной системы для составления маршрута в сети общественного транспорта г. Перми с системой пересадок.

Объектом исследования является система составления маршрутов в сети общественного транспорта.

Предметом исследования является автоматизация построения оптимального маршрута в сети общественного транспорта с учетом пересадок.

Для моделирования системы используются функциональный и объектно-ориентированный подходы.

Разберем примеры некоторых существующих в настоящее время информационных систем, предоставляющих информацию и услуги для самостоятельных путешествий:

Яндекс Карты – поисково-информационная картографическая служба Яндекса. В том числе она представляет собой навигационное приложение для перемещения по городу без личного автомобиля. Одна из функций данной системы – это отображение движения общественного транспорта, а также построение маршрутов. Более углубленно, данная система предоставляет пользователю подробнейший маршрут из точки А в точку Б, включая всевозможные варианты передвижения на транспорте.

Гугл карты – набор приложений, построенных на основе бесплатного картографического сервиса и технологии, предоставляемых компанией Google. Сервис представляет собой карту и спутниковые снимки планеты Земля. Также с сервисом интегрирован бизнес-справочник и карта автомобильных дорог с поиском маршрутов

2gis – международная картографическая компания, выпускающая одноимённые электронные справочники с картами городов. Предоставляет такие услуги, как подробная карта городов России: поиск по адресу, телефоны, отзывы, фото, часы работы фирм и удобный поиск проезда.

К достоинствам всех рассмотренных систем можно отнести то, что в них присутствует функция геолокации пользователя, также каждая из систем рассчитывает время маршрута. Еще немаловажным достоинством является то, что каждая из систем поддерживает систему пересадок

К недостаткам можно отнести запутанный интерфейс, многие полезные функции сложно найти, так же на некоторых сайтах плохо оптимизирована скорость загрузки.

В целом, каждая из этих систем выполняет основную функцию это – построение маршрута с системой пересадок.

Система должна решать задачи – оптимизация управления системы пересадок, предоставления населению актуальной информации обо всех маршрутах, расписаниях и видах транспорта города Перми и составление оптимального пути из начальной точки в

конечную с использованием системы пересадок. Так же система должна иметь возможность делать и просматривать любимые маршруты и виды транспорта.

Целью информационной автоматизированной системы для составления маршрута в сети общественного транспорта с системой пересадок являются:

— оптимизация процесса формирования маршрута в сети общественного транспорта с системой пересадок;

— ускорение получения информации пользователем;

— улучшение качества принимаемых решений.

Назначением разрабатываемой системы является автоматизация процесса составления маршрута в сети общественного транспорта с системой пересадок в соответствии с запросами пользователя.

Для дальнейшего проектирования информационной системы следует выделить основных акторов, взаимодействующих с системой, и установить возможности для каждого из выделенных. (см. рис. 1, 2 и 3).

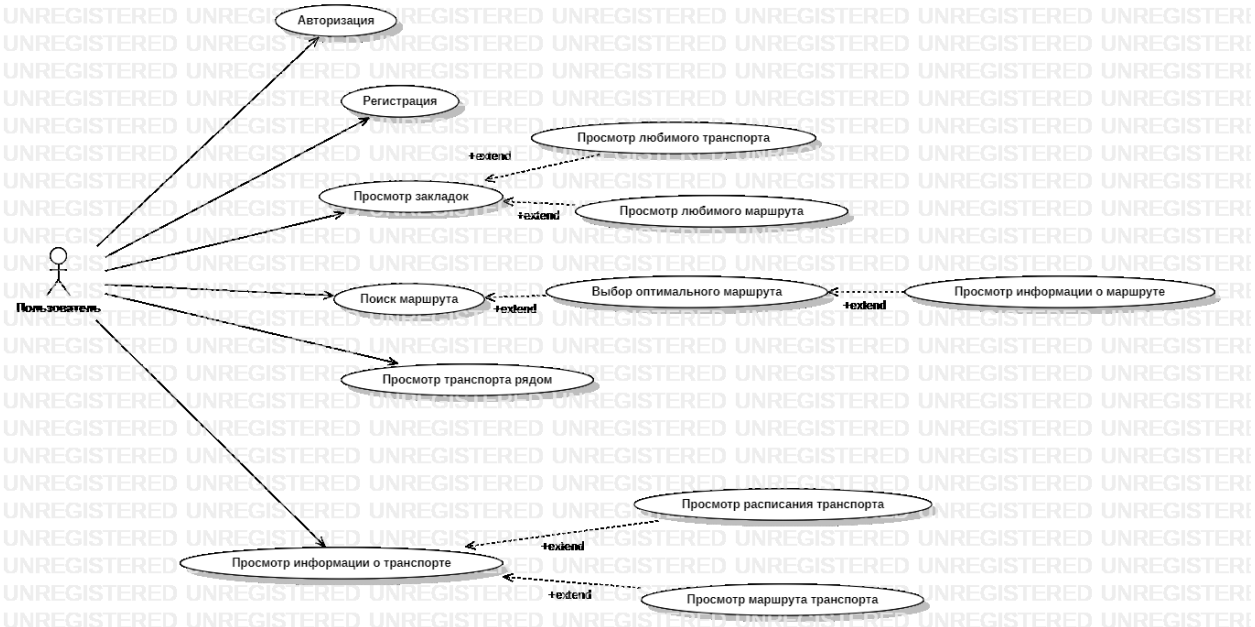


Рис. 1 Диаграмма прецедентов пользователя

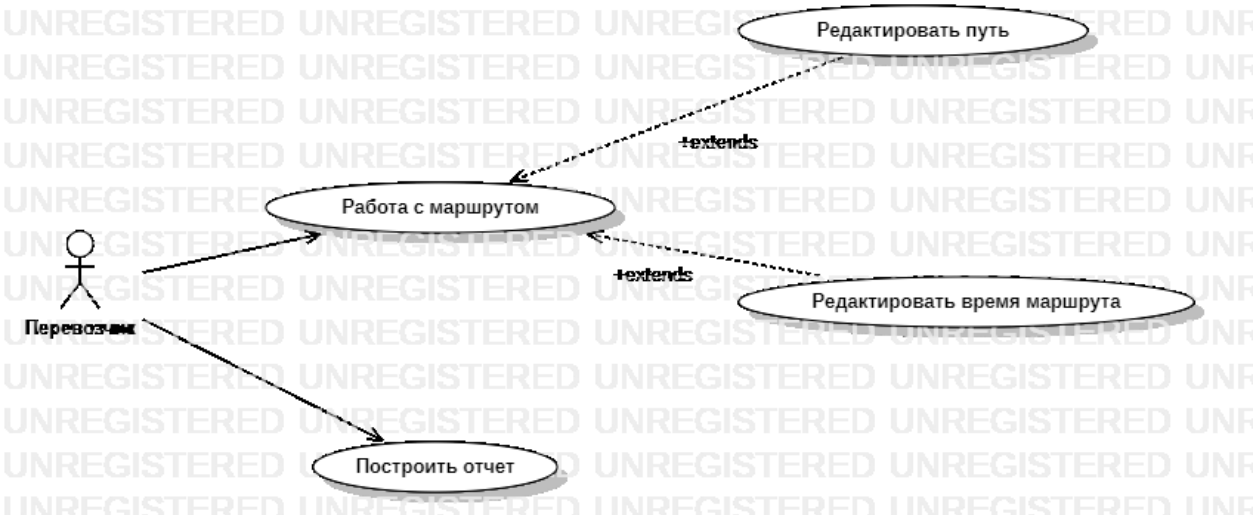


Рис. 2 Диаграмма прецедентов перевозчика

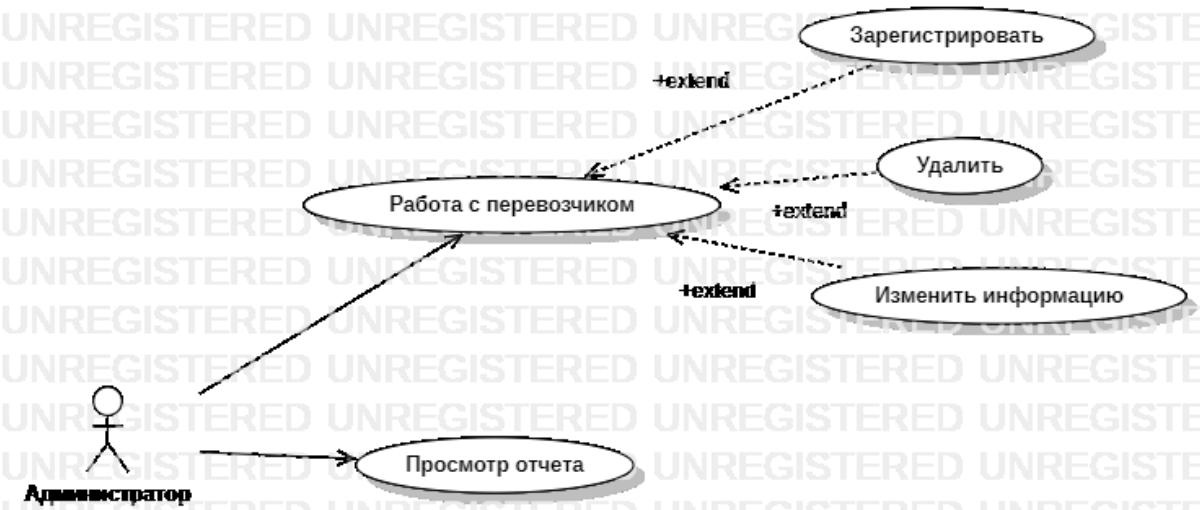


Рис. 3 Диаграмма прецедентов администратора

Для пользователей информационная система предоставляет возможность посмотреть расписание движения транспорта на интересующей остановке, путь, по которому идет транспорт и расписание движения для интересующего маршрута. За актуальностью информации о маршрутах следят перевозчики, которые записывают в информационную систему информацию обо всех изменениях их маршрута. Перевозчикам это необходимо для построения отчетов по их маршруту, которые требуют государственные органы, и для планирования маршрутов – система показывает загруженность транспортом различных районов города Перми.

Для построения кратчайшего пути пользователю необходимо выбрать остановки из списка (начальную точку и конечную точку), который появляется при вводе первых букв названия остановок, также пользователь может выбрать остановки на карте. Также пользователю нужно обозначить, нужны ли ему пересадки. Система автоматически построит таблицу с возможными путями, отсортировав их по времени. Пользователю остается выбрать оптимальный для него путь. Выбрав путь, система показывает следующую таблицу с подробной информацией о маршруте. Информация включает в себя: время следования, номер транспорта, в каком месте сделать пересадку и на какой рейс, если пользователь выбрал путь с пересадкой.

Путь без пересадок может не существовать. Если все же пользователь выбрал путь без пересадок, но такого пути нет, в таком случае, пользователю отображается сообщение об этом и система автоматически строит путь с пересадками.

В случае с пересадкой, остановка – это точка, в которой пользователь остановится во время маршрута. Алгоритм построения маршрута находит оптимальный путь передвижения по сети, соединяя нужные пользователю места остановок.

В информационной системе будет использоваться технология Drag-and-Drop. Данная технология позволяет перемещать маркеры остановок. При перемещении маркера будет строиться новый путь. Так же будет задействована технология Google Maps API. На ее основе можно создать удобный пользовательский диалог с достаточно широкими возможностям для поиска остановочных пунктов и просмотра найденного пути.

Процесс поиска и выбора маршрута изображен на рисунке 4

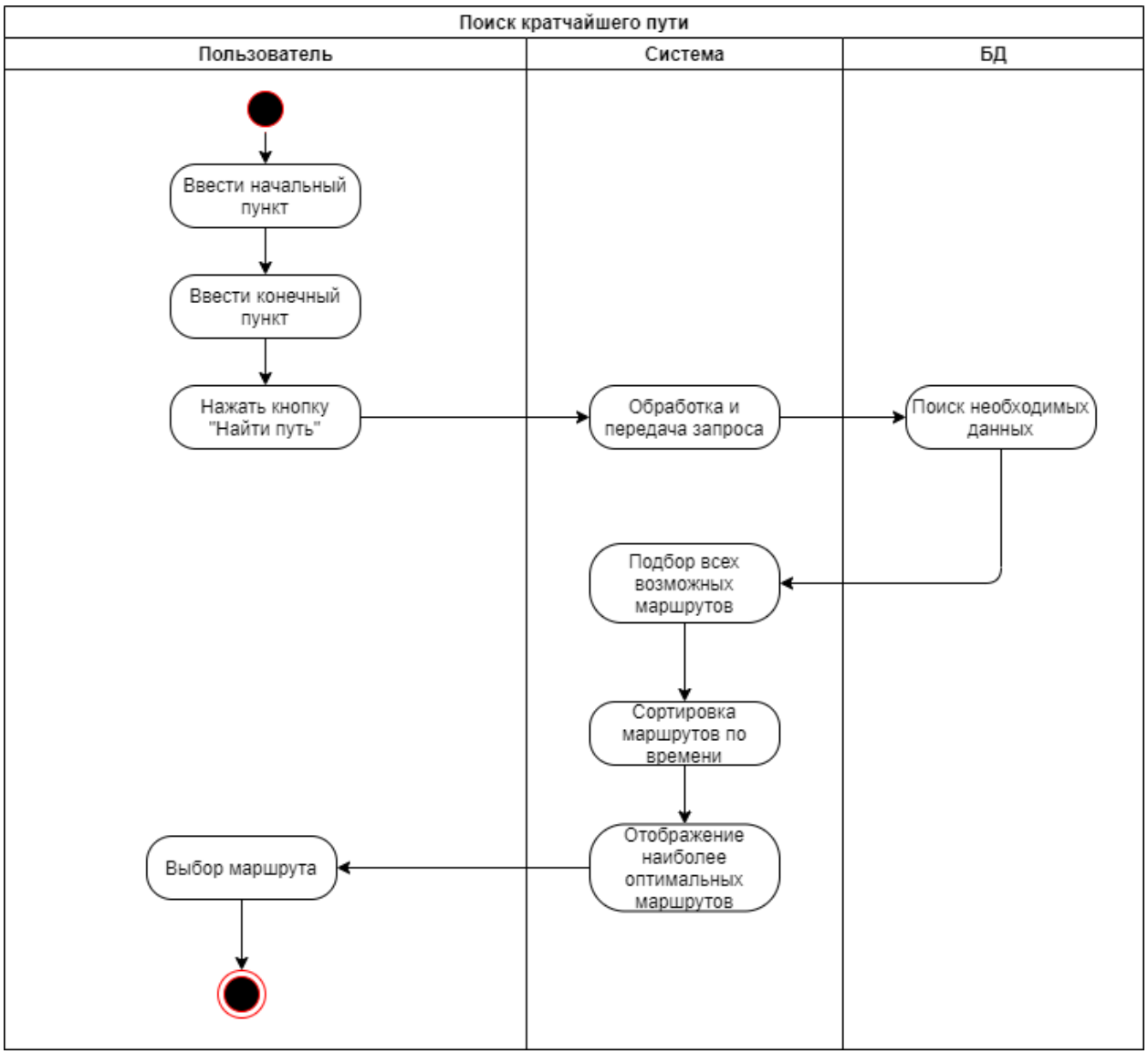


Рис. 4 Диаграмма деятельности выбора маршрута

Зайдя в отдельную панель приложения, пользователь может посмотреть полную информацию о нужном ему рейсе. Сперва панель дает выбрать вид транспорта. Нажав на интересующий пользователя вид транспорта, появляется таблица из всех существующих маршрутов данного вида транспорта. Для облегчения поиска, существует фильтрация по номеру рейса. Нажав на интересующий номер маршрута, пользователю предоставляется подробная информация о маршруте (путь следования, кто перевозчик, модель транспорта). Пользователь может посмотреть время прибытия рейса на определенной остановке, для этого ему нужно в открывшейся таблице нажать на “расписание”. Процесс показан на рисунке 5.

Также пользователь может воспользоваться данной функцией, нажав на определенную остановку, в таком случае, ему откроется таблица со всем транспортом, который проходит через данную остановку. Далее потребуется выбрать нужный номер рейса. Фактически, рейсы и остановки в таблицах являются ссылками.

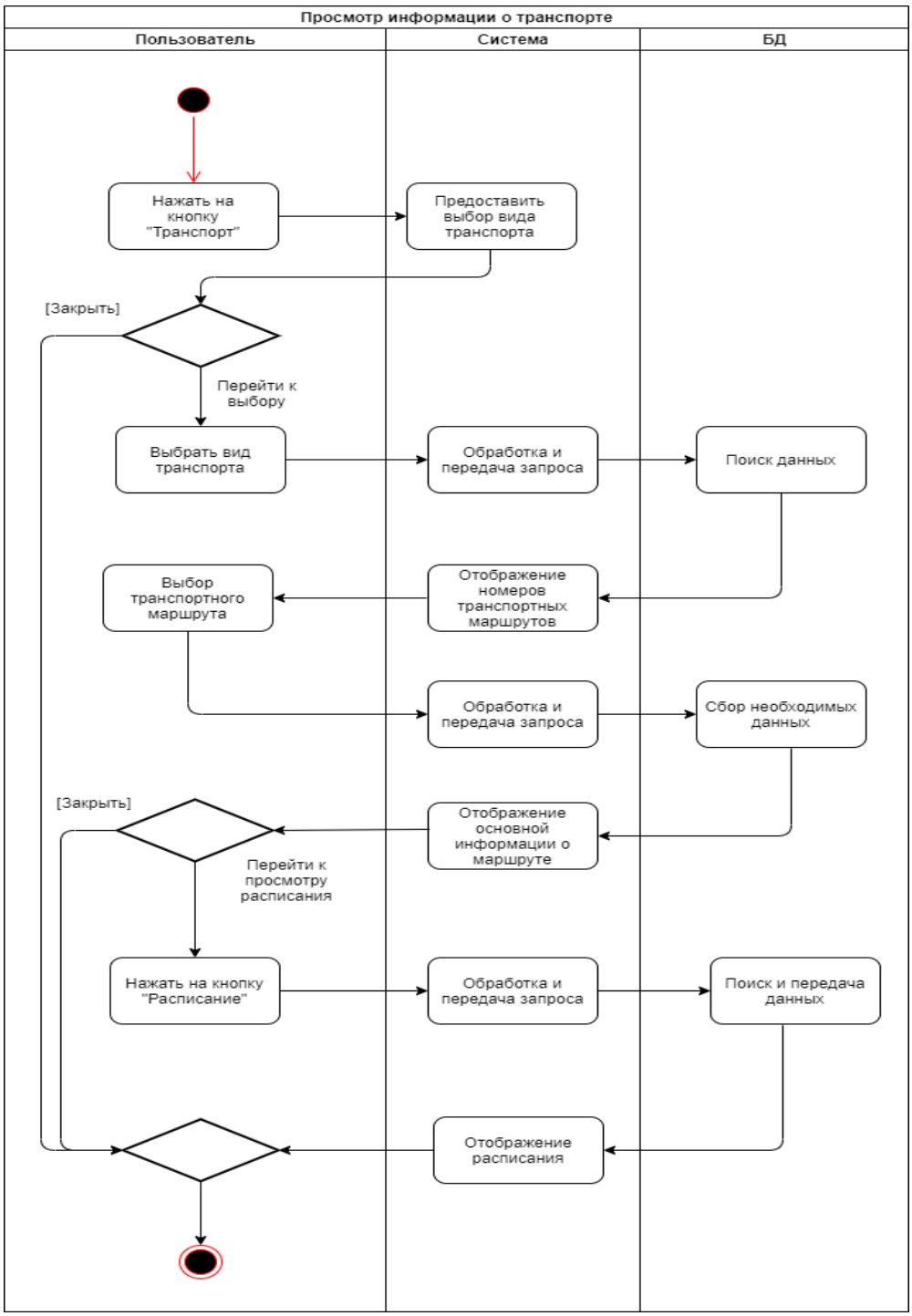


Рис. 5 Диаграмма деятельности просмотра информации о транспорте

В завершении проектирования информационной системы для построения маршрута в сети общественного транспорта г. Перми с системой пересадк был создан макет пользовательского интерфейса программы с помощью Figma. Рассмотрим прототипы интерфейса некоторых компонентов.

После входа в систему, пользователю открывается главная страница приложения (см. рис. 6). На ней расположена боковая панель, которая позволяет найти какое-либо место или адрес на карте. В центре расположена панель с иконками, которые представляют собой функции приложения. Первая иконка – построение маршрута, вторая – просмотр информации о транспорте, третья – просмотр закладок. Справа экрана расположен небольшой блок, который содержит в себе аватар пользователя, при нажатии на который будет возможность выйти из аккаунта.

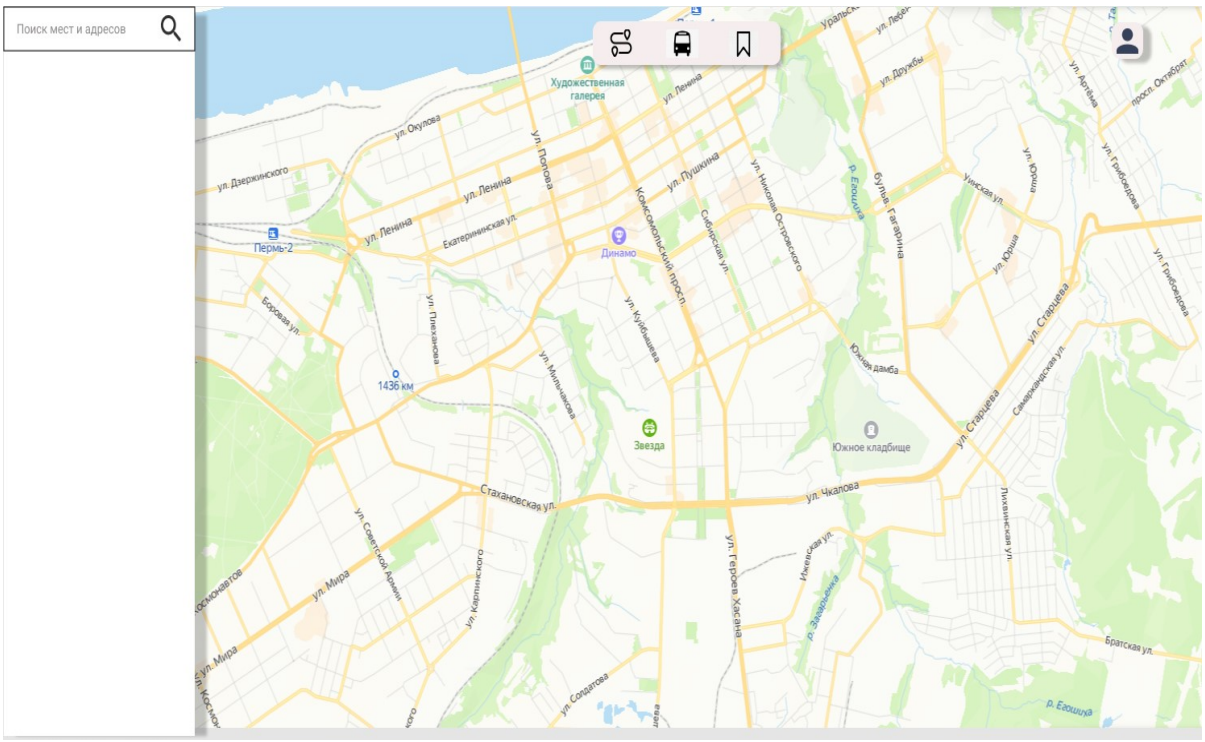


Рис.6 Прототип главной страницы

При нажатии на иконку построения маршрута, на боковой панели появляются необходимые настройки маршрута (см. рис. 7). Настройки маршрута включают в себя: начальную точку, конечную точку, выбор транспорта. Также есть возможность использовать любимый адрес или маршрут из закладок.

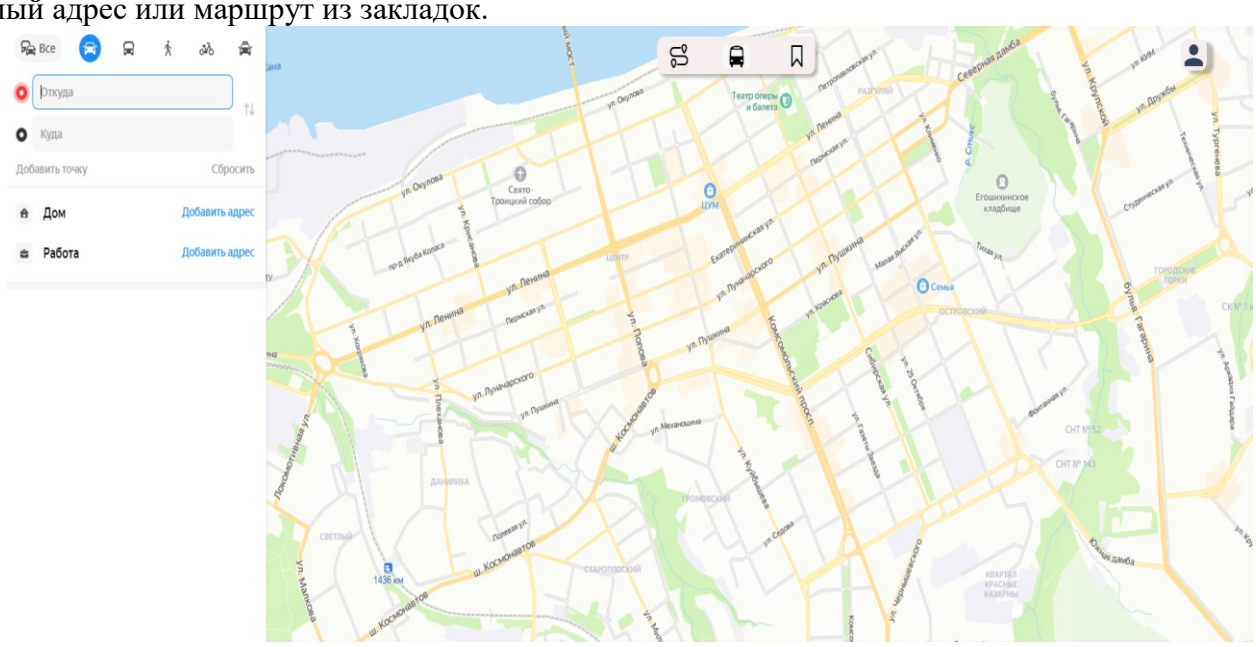


Рис. 7 Прототип страницы построения маршрута

Разработанная система предназначена для использования жителями города Перми, совершающими поездки на общественном транспорте. Система реализует:

- Отображение текущего местоположения пользователя;
- Подбор транспорта;
- Расчет времени маршрута;
- Расчет стоимости маршрута;
- Поиск адресов на карте;
- Построение маршрутов с возможностью пересадок.

Как итог система позволяет сэкономить время жителей города Перми.

Библиографический список

1. Оптимизация составления маршрутов общественного транспорта при создании автоматизированной системы поддержки принятия решений. // Е.А. Кочегурова, Ю.А. Мартынова [Электронный ресурс] URL: https://cyberleninka.ru/article/n/optimizatsiya-sostavleniya-marshrutov-obschestvennogo-transporta-pri-sozdanii-avtomatizirovannoy-sistemy-podderzhki-prinyatiya/viewer (дата обращения: 10.05.2021).
2. Пассажирские перевозки // Муниципальное образование город Пермь.

DESIGN AND DOCUMENTATION OF THE INFORMATION SYSTEM FOR MAKING A ROUTE IN THE PUBLIC TRANSPORT NETWORK OF PERM WITH A TRANSFER SYSTEM

Blaginykh Nikita V.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, salenctawer@gmail.com

This article discusses a system that helps residents of the city of Perm to compose a route from the starting point to the final point in the public transport network. The urgency of designing an automated system for planning a route in a public transport network with a transfer system has been substantiated. The analysis of similar information systems is made. The functionality of the system has been determined. The choice of a database management system has been made. The diagram of use cases, the diagram of actions are built, the prototype of the system interface is created.

Key words: route, public transport network, selection of transport, route building, automate, system model, transfer system.

УДК 374

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ИНТЕРАКТИВНОГО ОБУЧЕНИЯ FRONT-END РАЗРАБОТКЕ

Игнатова Анна Алексеевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, missjellx@gmail.com

Рассматриваются основные средства для проектирования и документирования информационной системы интерактивного обучения Front-end разработке. В ходе исследования [1] было выявлено, что обучение должно следовать принципам программируемого обучения [2]. В результате анализа аналогов информационных систем обучения Front-end разработке, были представлены основные требования к нашей системе: комфортный в использовании интерфейс, интерактивность, возможность обучения пользователей с разным уровнем знаний, системность и последовательность изложения материала, предоставление пользователям возможности создания веб-приложений. В результате анализа средств проектирования и разработки было выбрано инструментальное ПО для проектирования системы обучения: CASE-средство StarUML для моделирования системы [3]; кроссплатформенный онлайн-сервис Figma для разработки прототипа интерфейса приложения; для разработки клиентской части приложения был выбран язык гипертекстовой разметки HTML, каскадная таблица стилей CSS, язык программирования

JavaScript, так как согласно статистике [4], этот язык является самым популярным для разработки клиентской части веб-приложений и JavaScript-библиотека React; для разработки серверной части приложения был выбран язык Python и фреймворк Django из-за использования их во многих компаниях [5] и ввиду наличия подробной документации; в качестве СУБД для нашего приложения была выбрана MySQL из-за простоты освоения и гибкой настройки [6]. В результате проектирования информационной системы обучения Front-end разработке мы описали поведение системы и взаимодействие с ней пользователя с помощью диаграмм прецедентов, деятельности, последовательности и развертывания, и описали структуру будущего приложения с помощью диаграммы классов. На основе диаграмм мы построили примерный прототип интерфейса, изобразив весь функционал, который был представлен ранее в диаграммах.

Ключевые слова: средства для проектирования и документирования, Front-end разработка, интерактивность, создание веб-приложений, инструментальное ПО, CASE-средство, клиентская часть приложения, серверная часть приложения, диаграмма прецедентов, диаграмма деятельности, диаграмма последовательности, диаграмма развертывания, диаграмма классов, прототип интерфейса.

Библиографический список

1. Хеннер Е.К. ВЫЧИСЛИТЕЛЬНОЕ МЫШЛЕНИЕ. Образование и наука. 2016; (2):18-33. [сайт] URL: https://doi.org/10.17853/1994-5639-2016-2-18-33 (дата обращения 29.01.2021);
2. Скиннер Б. Наука учения и искусство преподавания: [сайт] URL: https://www.yumpu.com/xx/document/read/46790147/- (дата обращения 29.01.2021);
3. Национальный Открытый Университет «ИНТУИТ» – Введение в UML, обзор CASE-средств для построения диаграмм UML: [сайт] URL: https://intuit.ru/studies/professional_retraining/941/courses/229/lecture/5963 (дата обращения 29.01.2021);
4. Usage statistics of JavaScript as client-side programming language on websites: [сайт] URL: https://w3techs.com/technologies/details/cp-javascript (дата обращения 10.03.2021);
5. Quora: Which companies are using the Python language: [сайт] URL: Which companies are using the Python language? – Quora (дата обращения 23.03.2021);
6. MySQL – open source database: [сайт] URL: MySQL (дата обращения 10.04.2021);

DESIGN AND DOCUMENTATION OF THE INFORMATION SYSTEM OF INTERACTIVE LEARNING FRONT-END DEVELOPMENT

Ignatova Anna A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, ivanov@email.ru

Abstract. There are considered the main tools for designing and documenting an information system of interactive learning Front-end development. During the research [1], it was found that learning should follow the principles of programmed learning [2]. As a result of the analysis of analogs of information systems of learning Front-end development, the main requirements for our system were presented: a user-friendly interface, interactivity, the ability to learn users with different levels of knowledge, consistency of presentation of the material and to provide for users the ability to create web applications. As a result of the analysis of design and development tools, were chosen the following instrumental software for designing the learning system: CASE-tool StarUML for modeling system [3]; cross-platform online service Figma for developing an application interface prototype; for the development of the client-side of the application were chosen: the hypertext markup language HTML, the cascading style sheet CSS, the JavaScript programming language, because according to statistics [4], this language is the most popular for

the development of the client-side of web applications and the JavaScript library React; for the development of the server side of the application were chosen the Python language and the Django framework due to their usage in many companies [5] and due to the availability of detailed documentation; MySQL was chosen as the DBMS for our application due to its flexible configuration [6]. As a result of designing an information system for learning Front-end development, we described the behavior of the system and the interaction with it by the use-case, activities, sequence and deployment diagrams, and described the structure of the future application using a class diagram. Based on the diagrams, we built an approximate prototype of the interface, with all the functionality that was presented earlier in the diagrams.

Keywords: design and documentation tools, front-end development, interactivity, web-application creation, software, CASE-tool, front-end, back-end, use-case diagram, activity diagram, sequence diagram, deployment diagram, class diagram , interface prototype.

УДК 004.415

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ СОЗДАНИЯ ИНДИВИДУАЛЬНОГО ТУРИСТИЧЕСКОГО МАРШРУТА

Сычев Иван Андреевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, sychov.ivan2000@yandex.ru

В статье рассмотрена система, предназначенная для получения необходимой и актуальной информации, упрощения планирования путешествий с учетом индивидуальных пожеланий пользователя. Обоснована актуальность проектирования автоматизированной системы создания индивидуального туристического маршрута. Сделан анализ некоторых информационных систем, предоставляющих информацию и услуги для самостоятельных путешествий. Определены функциональные возможности системы. В системе планируется автоматизировать поиск транспорта на весь маршрут путешествия, а также выстраивание точек маршрута в оптимальном порядке, так же в системе предусмотрена геолокация и возможность отслеживания передвижения по маршруту доверенным лицам и другие необходимые функции. Сделан выбор системы управления базой данных. Построены функциональная модель процесса создания индивидуального маршрута, диаграмма вариантов использования, диаграмма действий, создан прототип интерфейса системы.

Ключевые слова: индивидуальный туристический маршрут, подбор транспорта, построение маршрута, автоматизировать, модель системы, графы.

В России с ее богатым культурным наследием и обширной территорией достаточно много различных уникальных исторических, природных, культурных и других объектов. Многие люди путешествуют по стране на личном или общественном транспорте. Например, в рамках конференции посвященной турпотенциалу Пермского края руководитель Агентства по туризму и молодежной политики Пермского края подвела итоги 2020 года: «В прошлом году Пермский край принял 754 тысячи туристов. А по данным аналитики сотовых операторов – 2,3 миллиона туристов, то есть в три раза больше данных официальной статистики» [1].

Самостоятельное планирование путешествий является сложным и затратным по времени мероприятием, особенно если путешествие планируется на общественном транспорте. Во время путешествий возможны неожиданные ситуации, когда хотелось увидеть какую-то достопримечательность или место, а забыли, уехали далеко, а возвращаться нет времени. Поэтому к путешествию лучше готовиться заранее, при этом, не хочется тратить лишнее время.

В сфере туризма, как и в других сферах, все больше используется автоматизация, например А. М. Ветитнев считает что «Современная индустрия туризма является глобальным компьютеризированным бизнесом, в котором пересекаются интересы туристических корпораций, транспортных компаний и гостиничных цепочек всего мира.»[2].

Актуальность данной темы состоит в том, что существует множество готовых стандартных туристических маршрутов, но они не всегда могут удовлетворять потребностям туристов, многие по тем или иным причинам путешествуют и составляют маршруты самостоятельно.

Цель данной работы состоит в проектировании информационной автоматизированной системы создания индивидуального туристического маршрута.

Объектом исследования является система составления туристических маршрутов.

Предметом исследования является автоматизация построения оптимального туристического маршрута с учетом различных видов транспорта.

В работе использовались системный подход, метод анализа, сравнения.

Для моделирования системы используются функциональный и объектно-ориентированный подходы.

Разберем примеры некоторых существующих в настоящее время информационных систем, предоставляющих информацию и услуги для самостоятельных путешествий:

YouRoute – система составления и планирования уникальных поездок по всему миру. К достоинствам сервиса можно отнести построение маршрута на конкретные даты с выбором желаемых для посещения стран и городов, расчёт стоимости всего путешествия, времени в пути, длины маршрута, выбор интересующих мест, услуг, вариантов досуга, бронирование отелей и ресторанов, бронирование автомобилей и другой техники через пункты проката, скачивание запланированной поездки в формате pdf.

Smorodina.com – путеводитель по России, с возможностью выбора экскурсий, интересных мест, точек общепита, вариантов досуга, бронирование отелей, авиабилетов.

Waytips.com – система планирования путешествий в любую точку мира. Сервис упорядочивает и автоматически разбивает план посещения достопримечательностей на несколько дней, имеется возможность вести блог, присутствуют готовые путешествия, предоставляется возможность подбора авиабилетов.

К недостаткам анализируемых сервисов можно отнести то, что они не могут построить весь маршрут от начальной точки до точки возврата с учетом различного транспорта. Так же к недостаткам можно отнести запутанный интерфейс, многие полезные функции сложно найти, так же на некоторых сайтах плохо оптимизирована скорость загрузки.

Проектируемая система должна позволять выбрать тип маршрута для более оптимальных рекомендаций, с учетом этого выбрать объекты для посещения и необходимые услуги, а также оплатить или забронировать их, построить оптимальный маршрут. Необходима возможность поделиться маршрутом, например с близкими людьми, которые смогут отслеживать его прохождение. Так же система должна иметь возможность отмечать пройденные точки маршрута с учетом геолокации или вручную. В тоже время система должна предоставлять достоверную и актуальную информацию услуг, местоположение точек предоставления этих услуг, поэтому необходима возможность внесения дополнений пользователями.

Целью информационной автоматизированной системы создания индивидуального туристического маршрута являются:

оптимизация процесса формирования индивидуального туристического маршрута;

ускорение получения информации пользователем;

— улучшение качества принимаемых решений.

Назначением разрабатываемой системы является автоматизация процесса составления индивидуального туристического маршрута в соответствии с запросами пользователя.

В начале для моделирования процесса создания индивидуального туристического маршрута применим методологию IDEF0. При моделировании процесса рассматривается упрощенная версия системы, реализующая только базовые функции (см. рис. 1).

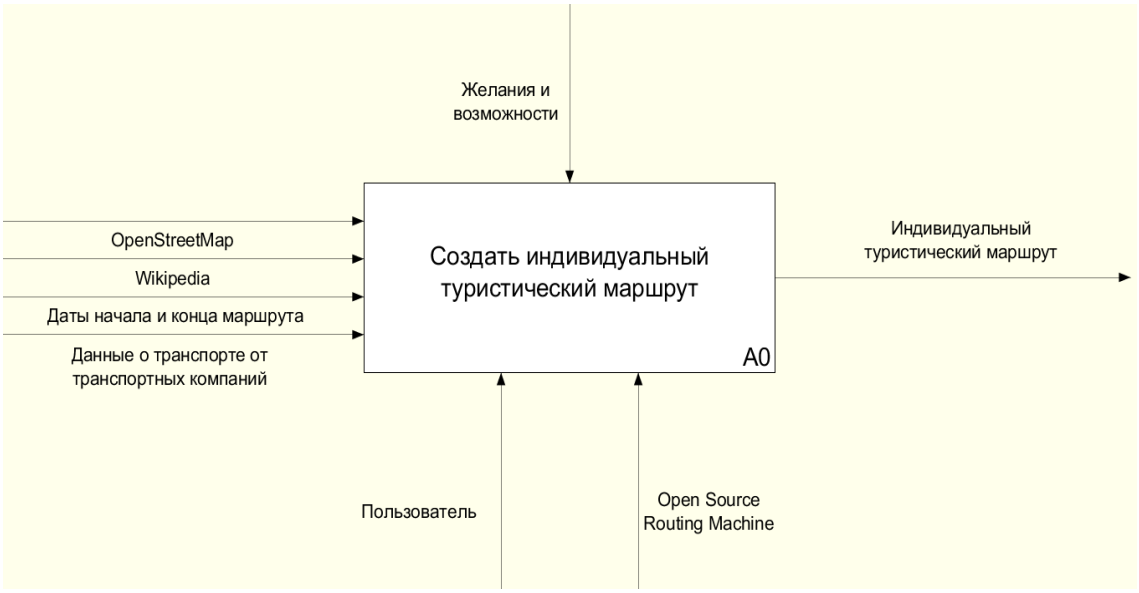


Рис. 1 Начальная контекстная диаграмма функциональной модели

Создание собственной базы достопримечательностей является трудоемким и длительным процессом, в связи с этим система может получать геоданные достопримечательностей и других объектов из некоммерческого веб-картографического проекта OpenStreetMap. Поскольку данный сервис содержит только краткую информацию более подробное описание объектов и фото берется из общедоступной интернет-энциклопедии Wikipedia. Для подбора транспорта между объектами используются данные из различных источников, таких как сайты железнодорожных компаний, авиакомпаний, сайты городского и междугороднего общественного транспорта и т.д. поскольку не была найдена единая база. Выбор достопримечательностей и объектов осуществляется пользователем.

Блок создания индивидуального туристического маршрута разбивается на 3 модуля: «выбрать достопримечательности», «подобрать транспорт», «построить маршрут» (см. рис. 2).

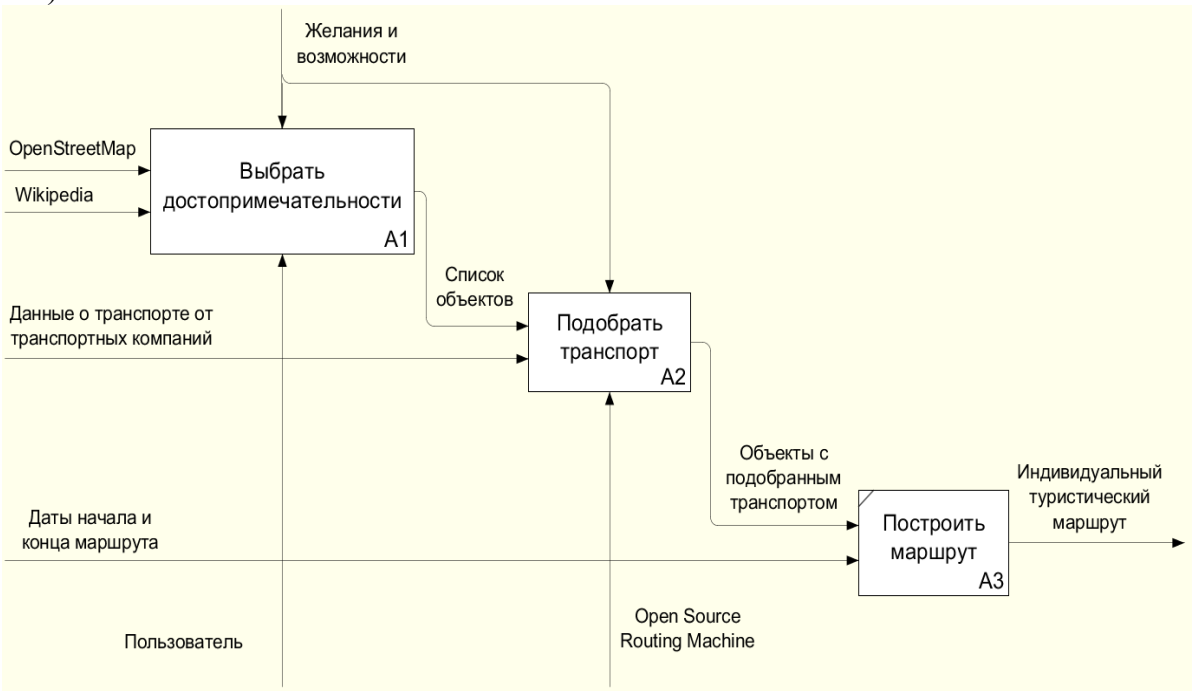


Рис. 2 Декомпозиция блока «Создать индивидуальный туристический маршрут»

Модуль выбора достопримечательностей обрабатывает информацию и взаимодействует с пользователем для создания списка объектов для посещения.

Модуль подбора транспорта рассчитывает расстояние и находит варианты передвижения между точками маршрута, затем происходит фильтрация вариантов транспорта с учетом пожелания пользователя, в следующий блок передается список точек маршрута и подходящие варианты передвижения между ними.

Далее модуль построения маршрута обрабатывает полученный список и выстраивает их в оптимальном порядке, основываясь на ранее собранной информации. Реализовать это можно построив граф и решив задачу поискать оптимального пути. Результатом обработки данной информации является туристический продукт, содержащий расписание поездки, транспортные расходы, данные о посещаемых объектах.

Рассмотрим диаграмму вариантов использования моделируемой системы (см. рис. 3).

Рис. 3 Диаграмма вариантов использования

В системе присутствует два актора: турист и доверенное лицо, с которым турист поделился маршрутом. Турист имеет доступ ко всем функциям системы, главные из которых создание, просмотр, следование по маршруту. Доверенное лицо может просматривать маршрут и получать всю доступною информацию по нему.

Построение маршрута пользователем заключается в взаимодействии с модулем создания маршрута. Вначале выбирается тип маршрута. С учетом типа путешествия система предлагает достопримечательности, места отдыха, услуги. Далее пользователь в

произвольном порядке выбирает даты поездки, виды транспорта и составляет список точек маршрута. (см. рис. 4).

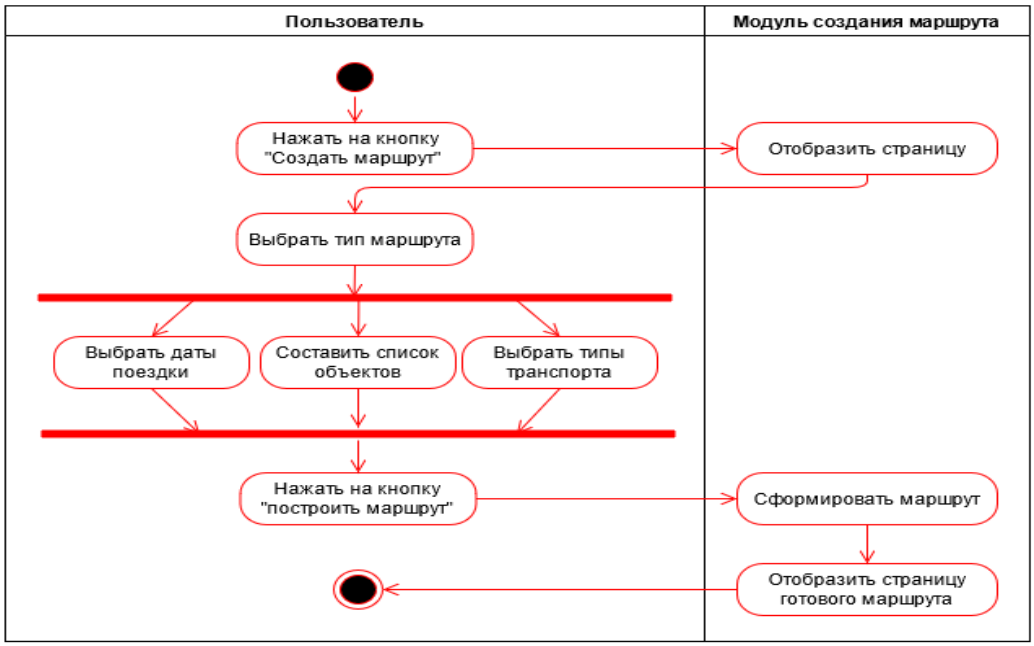


Рис. 4 Построение маршрута пользователем

Необходимым компонентом любой информационной системы является база данных. Поскольку система должна оперировать очень большим объемом слабоструктурированных данных для работы с ними лучше использовать NoSQL**.** Для разрабатываемой системы лучше всего подходит графовое хранилище, поскольку одной из главных задач системы является построение маршрута и подбор объектов исходя из многочисленных связей и параметров.

Для успешного внедрения веб-приложения в эксплуатацию необходим удобный, интуитивно понятный интерфейс поскольку от него зависит желание пользователя взаимодействовать с системой (см. рис. 5).

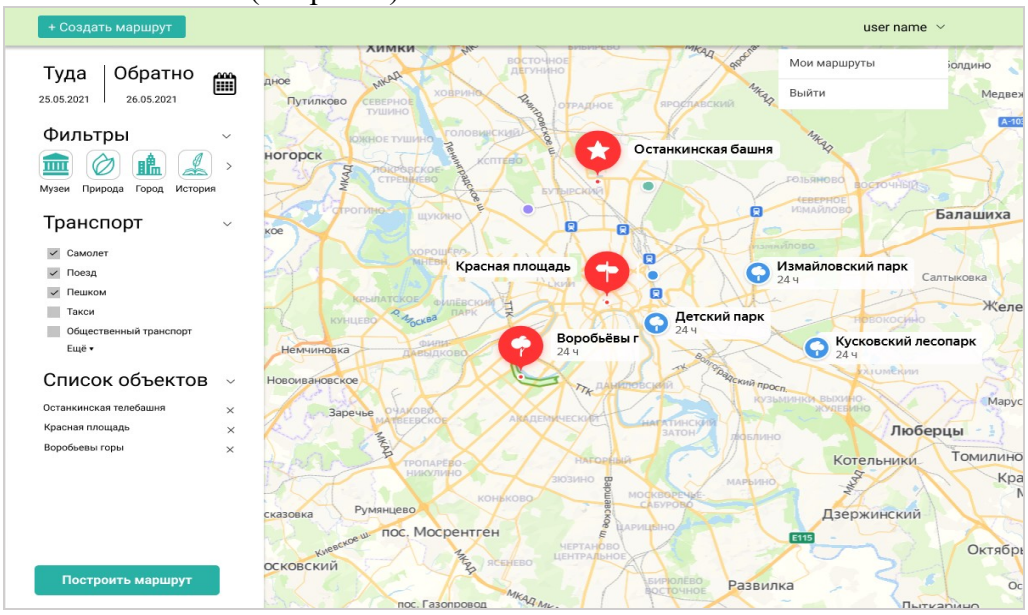


Рис.5 Прототип страницы построения маршрута

При переходе на страницу построения маршрута появляется модальное окно с выбором типа маршрута, на самой странице построения маршрута находиться карта и боковое меню, содержащие фильтры, которые могут быть применены к карте, форму выбора транспорта, список выбранных объектов, поля выбора желаемых дат начала и конца маршрута и кнопку «построить маршрут».

При нажатии на метку на карте появляется модально окно с подробной информацией об объекте и фото. После нажатия на кнопку «построить маршрут», пользователь перенаправляется на страницу готового маршрута (см. рис. 6).

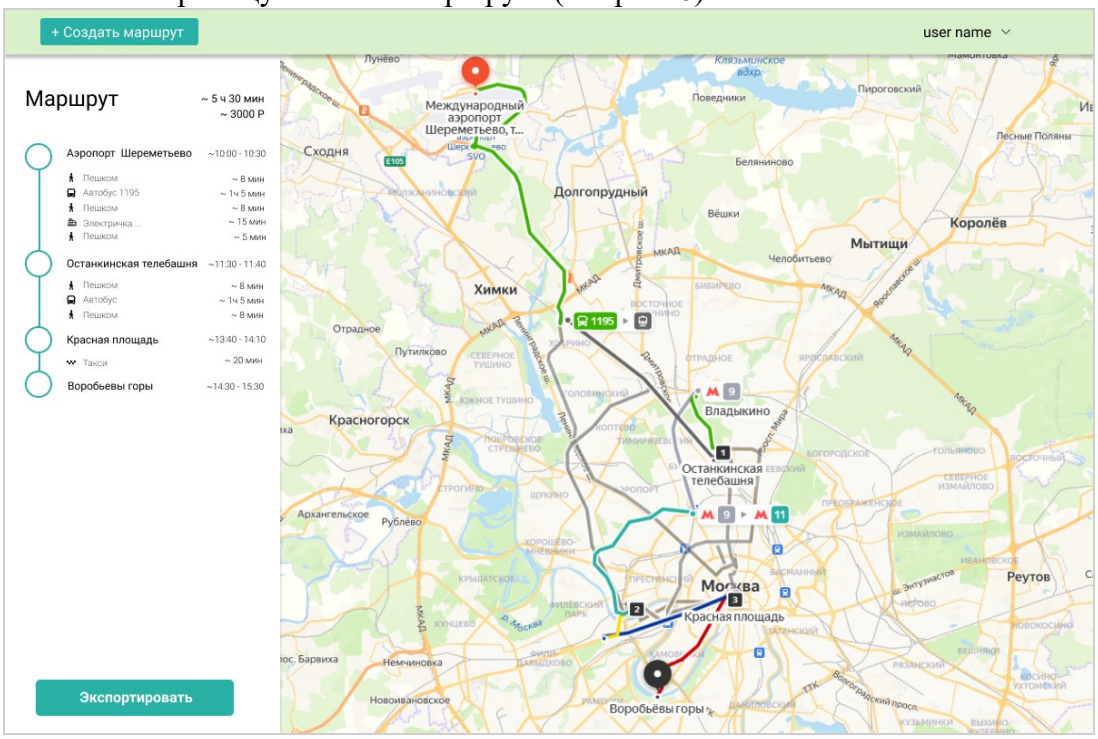


Рис. 6 Прототип страницы готового маршрута

Разработанная система индивидуальных туристических маршрутов предназначена для использования частными лицами при составлении собственных туристических маршрутов. В Системе предусмотрена возможность просмотра доверенными людьми местоположения туриста и последней отмеченной точки маршрута. Система реализует:

- упорядочивание точек маршрута в оптимальном порядке;
- подбор оптимальный транспорта между точками маршрута;
- расчет стоимости поездки;
- расчет времени прохождения по выбранному маршруту;
- поиск достопримечательностей, и других объектов;
- просмотр полной информации об объектах и возможность внесения своих уточнений;
- поиск рекомендаций сопутствующих услуг по пути следования;
- поиск, бронирование, оплату гостиниц, транспорта и других услуг;
- отметку пройдённых участков маршрута с учетом геолокации;
- подбор страховки.

Как итог система позволяет организовать наиболее эффективное управление временем пользователя.

Библиографический список

1. *Пермский* край подвел туристические итоги 2020 г. и озвучил планы на 2021г. // Ассоциация Туроператоров. [Электронный ресурс] URL: https://www.atorus.ru/news/press-centre/new/53589.html (дата обращения: 04.05.2021).
2. *Ветитнев, А. М.* Информационные технологии в туристской индустрии : учебник для академического бакалавриата // А. М. Ветитнев, В. В. Коваленко. – 2-е изд., испр. и доп. – М. : Издательство Юрайт, 2018. – 340 с.

DESIGN OF AN INFORMATION AUTOMATED SYSTEM FOR CREATING AN INDIVIDUAL TOURIST ROUTE

Sychev Ivan A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, sychov.ivan2000@yandex.ru

The article describes a system designed to obtain the necessary and up-to-date information, simplify travel planning, taking into account the individual wishes of the user. The relevance of designing an automated system for creating an individual tourist route is justified. The analysis of some information systems providing information and services for independent travel is made. The functional capabilities of the system are defined. The system is planned to automate the search for transport for the entire travel route, as well as the alignment of route points in the optimal order, as well as the system provides geolocation and the ability to track movement along the route to trusted persons and other necessary functions. The choice of the database management system is made. A functional model of the process of creating an individual route, a diagram of use cases, an action diagram, and a prototype of the system interface are constructed.

Keywords: individual tourist route, selection of transport, route construction, automate, system model, graphs.

УДК 004.415

ПРОЕКТИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ОЦЕНКИ КРЕДИТОСПОСОБНОСТИ ФИЗИЧЕСКИХ ЛИЦ

Меркушева Мария Сергеевна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, amequi12@gmail.com

В статье рассматривается актуальная проблема оценки кредитоспособности физических лиц. Нет сомнения в том, что с постоянным развитием информационных технологий, возрастает спрос на автоматизацию банковских услуг. Также, с внедрением информационных систем оценки, кредитные организации ожидают уменьшить банковские риски. В данной работе описана спроектированная модель информационной системы оценки кредитоспособности физических лиц.

Ключевые слова: банк, кредит, оценка кредитоспособности, скоринг, скоринговая система, скоринговая модель.

Введение

Кредит относится к числу важнейших категорий финансовых отношений и как экономическое явление играет уникальную роль в общественной жизни. И без того высокий уровень риска банковской деятельности увеличился под влиянием кризисных экономических условий, поскольку успешное функционирование банка в основном связано с результатами деятельности его клиентов.

Понятие кредитоспособности заёмщика, характеризующее способность физического лица полностью и в срок рассчитаться по своим долговым обязательствам, является ключевым фактором при принятии решений о целесообразности выдачи банковского

кредита заёмщику. Проблема оценки кредитоспособности физических лиц является одной из наиболее актуальных и дискуссионных тем на сегодняшний день.

Анализ предметной области

Нет сомнения в том, что с постоянным развитием информационных технологий, возрастает спрос на автоматизацию банковских услуг. Информационная система оценки кредитоспособности включает в себя взаимосвязанные элементы, обеспечивающие процесс принятия решения о предоставлении кредита заёмщику. В самой упрощенной форме скоринговая модель системы представляет собой взвешенную сумму определённых характеристик. В результате суммирования получается интегральный показатель – чем он выше, тем выше надёжность клиента[1]. После прохождения процедуры клиенту присваивается рейтинг платежеспособности. В зависимости от продукта и требований определяется, предоставлять кредит заёмщику или нет.

В более широком смысле скоринговая система изнутри представляет собой сложную систему автоматизации выдачи потребительских кредитов в отделениях банка. При этом универсальные аналитические инструменты используются в качестве ядра, с помощью которого также можно создавать свои собственные скоринговые модели. Единой системы оценки кредитоспособности не существует. Каждый банк использует либо одну из типовых программ, либо собственные методы. Критерии оценки платежеспособности заёмщиков в разных учреждениях также различаются. Параметры оценки для различных банковских продуктов тоже различны. Невозможно узнать, по какому именно методу клиента оценивает кредитная организация – эта информация строго для внутреннего использования[2].

К безусловным преимуществам скоринговых систем относятся[3]:

- Оптимизация стоимости рассмотрения заявки за счет автоматизации процесса принятия решений и выдачи кредита;
- Сокращение времени, необходимого для рассмотрения заявки, увеличение числа и скорости обработки заявок;
- Отсутствие субъективного мнения эксперта при принятии решения о выдаче кредита;
- Возможное снижение затрат;
- Минимизация рисков за счет автоматизации процесса принятия решения о выдаче кредита.

К недостаткам относятся:

- Субъективное мнение специалиста также влияет на оценку параметров и конечный результат, т.е. итоговый балл;
- Системы оценки нуждаются в постоянном обновлении. При разработке необходимо опираться на данные о последних клиентах;
- В основе выборки люди, которые получили кредит, нет информации о тех, кому отказали;
- Социально-экономические условия постоянно меняются;
- Программа оценивает не реального человека, а данные, которые он сообщает о себе, так, подготовленный клиент может представить о себе информацию таким образом, что практически гарантированно получит одобрение системы.

Требования

Сложность информационных систем, создаваемых в различных областях экономики, постоянно возрастает. Это объясняется тенденциями развития современных информационных технологий.

Информационную систему можно представить как приложение, написанное на основе клиент-серверной архитектуры. Использование данной архитектуры позволит создать надежное многопользовательское приложение, с централизованной базой данных и единой моделью оценки, не зависящее от аппаратной части сервера и клиентов.

Система должна содержать дифференцированные данные о потенциальных заёмщиках, для обеспечения максимально возможного уровня точности оценки, а также должна опираться на модули, имеющие собственный функционал.

Основные требования к системе:

1. Базирование на современных технологиях, быть построена на современных платформах (ОС и СУБД), позволяющих реализовать гибкость, открытость и масштабируемость системы;
2. Анализ кредитной истории и/или анкетных данных заёмщика;
3. Принятие решения с приемлемым уровнем предсказательной точности;
4. Удобная интерпретация, объяснимость и наглядность полученных результатов;
5. Быть понятной, легкой в эксплуатации и использовать современные технологии построения интерфейса;
6. Гибкость модели – возможность вносить коррективы в модель;
7. Ведение базы событий;
8. Наличие гибких возможностей настройки отчётов, доступных для применения обычными пользователями;
9. Обеспечение конфиденциальности, целостности и доступности информации;
10. Возможность классифицировать пользователей и предоставлять их различным категориям различные уровни доступа к системе и данным;
11. Обеспечение контроля за действиями пользователей на системном и прикладном уровне и их последующий анализ.

Основные модули информационной системы

Главная цель моделирования – поиск наилучших вариантов решений для разработки информационной системы. Чтобы успешно реализовать проект, то есть саму информационную систему, должно присутствовать его адекватное описание, а также должны быть построены непротиворечивые, полные информационные и функциональные модели информационной системы.

Функционал информационной системы был разбит на следующие блоки, которые представлены на общей диаграмме прецедентов (см. рис.1):

- Модуль обработки данных;
- Модуль управления выборкой данных;
- Модуль управления скоринговой моделью;
- Модуль управления отчётами;
- Центр обновлений;
- Модуль управления безопасностью;
- Модуль управления пользователями.

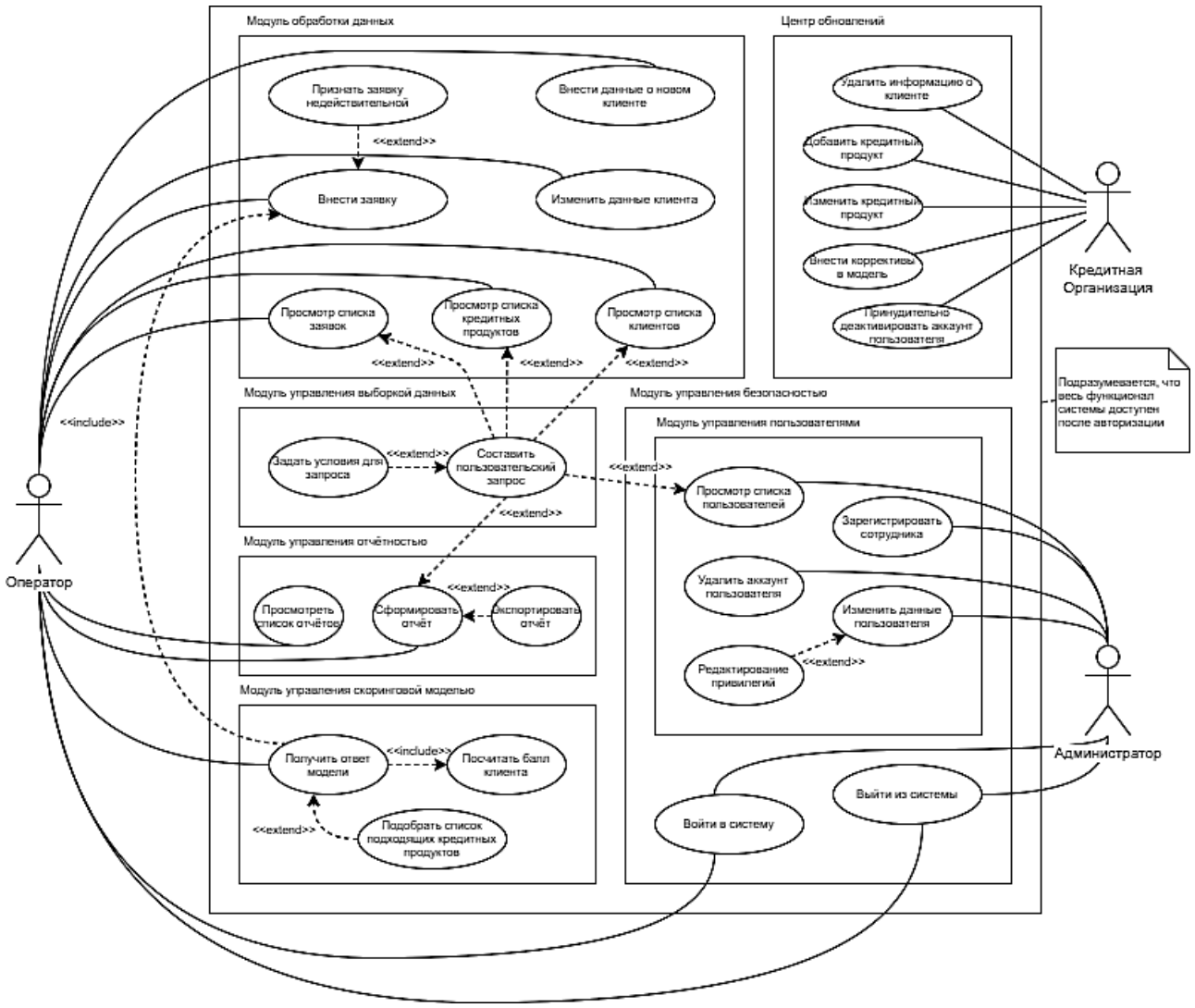


Рис. 1 – Общая диаграмма прецедентов

Согласно требованиям, информационная система должна включать в себя комплекс мероприятий, предотвращающий несанкционированный доступ и утечку информации. С этой целью необходимо ввести систему аутентификации – так у пользователя не будет доступа к основному функционалу, пока он не аутентифицируется в системе.

Также была введена ролевая модель контроля доступа. Данная модель безопасности определяет правила разграничения доступа к объектам информационной системы, основываясь на роли, присвоенной пользователю. Под ролью пользователя понимается набор прав субъекта по отношению к объектам.

Ещё один метод для обеспечения безопасности информационной системы – аудит действий пользователей. Использование этого инструмента позволяет снизить риск утечки информации по причине халатности пользователей. Все действия, совершенные пользователями, вне зависимости от их успешности, записываются в журнал событий. При активной журнализации событий можно отследить нестандартные действия пользователей или узнать причину сбоя системы. Для минимизации рисков, журнализацию действий необходимо вести как на стороне клиента и сервера, так и на стороне базы данных.

Модуль обработки данных необходим для получения и изменения данных, принимаемых из источников данных, а именно пользователей и базы данных. Этот модуль также предоставляет пользователям дополнительную поддержку при формировании отчётов. Модулю обработки данных предоставлен доступ к базе данных.

Как было сказано ранее, нередко клиенты обращаются в банк с целью получить кредит, однако прежде чем одобрять кредит физическим лицам, следует изучить их

финансовое положение. Для этого разрабатывается и используется математическая модель оценки кредитоспособности.

Выборка данных – это гибкий инструмент, который помогает пользователям работать с информацией. Модуль управления выборкой данных даёт пользователям возможность составлять собственные запросы для просмотра данных или создания отчётов. Посредством построения запросов пользователь может извлечь информацию из разных таблиц базы данных и собрать её для отображения в удобном виде. Задачи бывают разных типов, например, оператору необходимо найти информацию о том или ином клиенте, или получить список одобренных заявок за последний месяц для формирования отчётности. Из-за этого возникает потребность составлять запросы различной сложности, но использование фильтрации данных, применения различных условий и сортировки помогает упростить работу.

Некоторые необходимые операции недоступны для самостоятельного исполнения пользователям информационной системы. За их выполнение отвечает центр обновлений.

Для удобства работы с разрабатываемой системой, требуется спроектировать понятный пользовательский интерфейс. Пользовательский интерфейс – это внешний вид продукта, способ общения между пользователем и программой.

Разработанный макет интерфейса интуитивно понятен обычному пользователю, а также лёгок в эксплуатации (см. рис. 2).

Рис. 2 – Прототип интерфейса пользователя

Заключение

Использование скоринговых систем оценки позволяет снизить уровень невозвратов кредитов и обеспечивает более точную и менее трудозатратную оценку надёжности клиента.

Разработанная модель информационной системы позволяет провести анализ анкетных данных заёмщика и принять решение будет ли доступен запрашиваемый кредитный продукт клиенту.

Также модель описывает возможность построения и настройки отчётов. Учтена удобная интерпретация, объяснимость и наглядность полученных результатов. Были рассмотрены вопросы безопасности информационной системы, к которым относятся ведение базы событий, контроль за действиями пользователей и ролевая модель безопасности.

Библиографический список

1. Банкова, К. В. Использование скоринговых моделей для оценки кредитоспособности заемщиков в России / К. В. Банкова // Известия Академии управления: теория, стратегии, инновации. – 2011. – № 4. – С. 14-16.
2. Митчина, Т. Е. Методы оценки кредитоспособности заемщика (юридического лица) / Т. Е. Митчина, А. С. Деревянкина // Аллея науки. – 2018. – Т. 4. – № 5(21). – С. 72-76.
3. Сорокин, Я. В. Экономическое обоснование совершенствования оценки заемщика / Я. В. Сорокин, П. М. Зиятдинова // Устойчивое развитие науки и образования. – 2017. – № 6. – С. 104-109.

DEVELOPMENT AND DOCUMENTATION OF INDIVIDUAL CREDIT SCORE ASSESSMENT INFORMATION SYSTEM

Merkusheva Maria S.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, amequi12@gmail.com

The article deals with the current problem of assessing the creditworthiness of individuals. There is no doubt that with the constant development of information technologies, the demand for automation of banking services increases. Also, with the introduction of evaluation information systems, organizations expect to reduce bank risks. This article describes the development model of the information system for assessing the creditworthiness of individuals.

Keywords: bank, credit score, creditworthiness assessment, scoring, scoring system, scoring model.

УДК 004.5

ПРОБЛЕМА ПОДСЧЁТА СТОИМОСТИ ГОТОВОГО ПРОДУКТА У КОНДИТЕРОВ

Адутова Алина Ильнуровна

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, ivanov@email.ru

С древних времен кулинария была неотъемлемой частью жизни человечества. Люди каждой национальности создавали свои уникальные блюда и рецепты, увеличивая разнообразие вкусов мира. В наше время мы очень часто сталкиваемся с кондитерскими изделиями, в особенности с тортами. Большую часть данной продукции создают на производствах, но не всем легко найти то, что им нужно из-за различных особенностей организма и вкусовых предпочтений. Для создания изделий по индивидуальным предпочтениям, люди чаще всего обращаются к частным кондитерам. В случае увеличения размеров или изменениях формы готовой продукции возникает потребность в правильном расчёте количества ингредиентов и итоговой стоимости. В данной статье будут изучены инструменты для расчётов, а также целесообразность создания мобильного приложения.

Ключевые слова: перерасчёт ингредиентов, расчёт стоимости, торт, себестоимость.

Во все времена и во всех странах человечество постоянно создаёт что-то новое и необычное, пытаясь удивить остальных или решить насущную проблему. Так среди известных кондитерских изделий появился торт. Его форма и состав поражала воображение

не только на момент его появления, но и на данный момент. Ведь для частных кондитеров это является способом самовыражения.

Создание уникальных и вкусных тортов за частую связано с изготовлением по проработанной рецептуре, но с изменёнными размерами, формой и составом элементов торта. Для создания такого изделия приходится множество раз производить перерасчёт ингредиентов исходя из разницы с изначальными рецептами. Кроме того, правильный перерасчёт ингредиентов на прямую влияет на себестоимость торта.

Данная статья посвящена проблеме ценообразования готовых изделий. Материалом для данной статьи послужили выложенные в интернет распространённые проблемы начинающих кондитеров. Для анализа использовались различные блоги и каналы на таких социальных площадках, как ВК [1], YouTube [2] и Instagram [3]. Проанализировав отзывы кондитеров с данных ресурсов было выявлено, что большая часть кондитеров используют для расчётов готовые таблицы в Microsoft Excel [4] или создают их сами, и только малая часть используют для этого ручной подсчёт или мобильные приложения. Проверив удобство каждого вида используемых расчётов было выявлено множество особенностей каждого решения.

Таблицы в программе Microsoft Excel очень удобны в использовании с компьютеров и ноутбуков, обладая большим спектром настроек и лёгкостью изменения состава продукции. Неопытные пользователи для лучшей точности расчётов покупают такие таблицы у более опытных пользователей [5] или нанимают человека для работы с ними. Знакомые с этой программой кондитеры создают такие таблицы самостоятельно. Несмотря на наличие мобильной версии программы, создавать или редактировать большие таблицы на ней не удобно.

Кроме электронных таблиц существуют мобильные приложения и сайты для перерасчёта ингредиентов на новую форму и расчёта стоимости готового изделия. Однако в них существует ряд недоработок визуального и функционального характера.

Если брать стандартный ручной метод подсчёта, то такой подсчёт занимает довольно много времени и точность этих расчётов будет ниже подсчётов в таблицах программы Microsoft Excel.

Более подробно рассмотрим, что включает в себя себестоимость. Себестоимость – это затраты на производство продукции, выполнение работ или оказание услуг. Она рассчитывается из таких показателей как материальные затраты, оплата труда и налогов, амортизация [6].

Материальные затраты включают в себя стоимость ингредиентов, украшений, упаковки и коммунальных расходов. А также стоимость доставки в случае необходимости. Оплата труда состоит из сложности изделия, времени работы и профессиональный уровень кондитера. Эти три параметра составляют собой определённый процент от стоимости изделия, так называемые дивиденды. По мимо этого к стоимости готового изделия добавляется подоходный налог. Амортизация – это постепенное снижение ценности имущества в следствии его изнашивания. Так как кондитер не может работать без определённого оборудования добавляется небольшой процент к стоимости готовой продукции как плата за износ оборудования и гарант постоянного бесперебойного производства.

Перерасчёт ингредиентов включает в себя количество ингредиентов и их граммовку, а также форму которая была использована в рецепте и новую форму. Для выполнения перерасчёта требуется узнать площадь этих двух формы. После чего необходимо разделить площадь новой формы ($S_н$) на площадь текущей ($S_т$) для получения коэффициента разности площади (k). Полученный коэффициент необходимо перемножить с весом каждого ингредиента в рецепте [7].

$$k = \frac{S_н}{S_т}$$

где k – коэффициент разности площади
Sн – площадь новой формы
Sт – площадь текущей формы.

При производстве тортов используются формы круга и прямоугольника. Для изготовления сложных фигур по типу звезды изначально делают торт диаметром равным двум радиусам от центра изделия к его вершине. После чего приводится в задуманный вид. Площади круга (Sкр.) и прямоугольника (Sпр.) рассчитываются по следующим формулам:

$$S_{\text{кр.}} = \pi R^2$$

где Sкр. – площадь круга
π – постоянное число 3,14
R – радиус круга

$$S_{\text{пр.}} = a * b$$

где Sпр. – площадь прямоугольника
a – длина прямоугольника
b – ширина прямоугольника

Исходя из всего вышесказанного, автором статьи было предложено разработать прототип мобильного приложения, который решает следующие задачи:

1. Хранение, добавление, удаление и изменение данных для перерасчёта ингредиентов;
2. Фиксация стоимости трудозатрат;
3. Перерасчёт ингредиентов на круглую и прямоугольную формы разных размеров;
4. Расчёт полной стоимости торта;
5. Вывод результата в PDF файл.

Библиографический список.

1. Российская социальная площадка ВК [Электронный ресурс] URL: https://vk.com/feed (дата обращения: 05.06.2021).
2. Международная социальная площадка YouTube [Электронный ресурс] URL: https://www.youtube.com/ (дата обращения: 05.06.2021).
3. Международная социальная площадка Instagram [Электронный ресурс] URL: https://www.instagram.com/ (дата обращения: 05.06.2021).
4. Программа для работы с электронными таблицами Microsoft Excel [Электронный ресурс] URL: https://www.microsoft.com/ru-ru/microsoft-365/excel (дата обращения: 05.06.2021).
5. Готовая платна таблица для расчёта себестоимости [Электронный ресурс] URL: https://dessertcalculator.ru/#functions (дата обращения: 05.06.2021).
6. Особенности расчёта себестоимости торта [Электронный ресурс] URL: https://ardma.ru/marketing/osnovy-marketinga/744-kak-rasschitat-sebestoimost-torta/ (дата обращения: 06.06.2021).
7. Особенности расчёта ингредиентов для форм разных размеров [Электронный ресурс] URL: https://sladkiexroniki.ru/kak-pereschitat-ingredienty-dlya-torta-drugogo-diametra/ (дата обращения: 06.06.2021).

THE PROBLEM OF CALCULATING THE COST OF THE FINISHED PRODUCT IN CONFECTIONERS

Adutova Alina Ilnurovna

Perm State National Research University, 15, Bukireva st., Perm, 614990, Russia, ivanov@email.ru

Since ancient times, cooking has been an integral part of human life. People of every nationality have created their own unique dishes and recipes, increasing the variety of flavors of the world. Nowadays we are very often confronted with confectionery products, especially cakes. Most of these products are created in factories, but it is not easy for everyone to find what they need because of different body characteristics and taste preferences. To create products according to individual preferences, people most often turn to private confectioners. In the case of increasing the size or changing the shape of the finished product, there is a need to correctly calculate the number of ingredients and the final cost. This article will examine the calculation tools and the feasibility of a mobile app.

Keywords: recalculation of ingredients, cost calculation, cake, cost price.

УДК 004.09

ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «КОГНИТИВНЫЕ КАРТЫ КАК ИНСТРУМЕНТ ФОРМАЛИЗОВАННОГО ОПИСАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ»

Зрячих Владимир Андреевич

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, amfik0159@gmail.com

В данной статье рассматривается определение когнитивной карты, сравнительный анализ имеющихся информационных систем, обладающих нужным или схожим функционалом для построения когнитивных карт. Результаты представлены в сравнительной таблице. На основе анализа систем видно, что явного функционала для построения когнитивных карт нет. Общий функционал узлов и связей, который есть во всех рассмотренных системах, позволяет построить граф для когнитивных карт. Поэтому был сделан краткий обзор средств проектирования, а также описаны процесс разработки и компоненты информационной системы. Определены требования, которым должна удовлетворять разрабатываемая система. В ходе проектирования информационной системы были получены модели в виде диаграммы прецедентов, диаграммы последовательностей и диаграммы классов, удовлетворяющих требованиям для разрабатываемой системы. А также представлен прототип интерфейса.

Ключевые слова: проектирование информационной системы, когнитивная карта.

Когнитивная карта (карта познания, cognitive mapping) – это вид математической модели, представленной в виде графа и позволяющей описывать субъективное восприятие человеком или группой людей какого-либо сложного объекта, проблемы или функционирования системы.

В ходе выполнения выпускной квалификационной работы был проведен анализ информационных систем, обладающих нужным или схожим функционалом для построения когнитивных карт. Для анализа были выбраны следующие системы:

1. XMind 2020 [4];
2. Visual Understanding Environment (VUE) [3];
3. MindManager [2].

Исходя из анализа видно, что все приложения обладают похожими функциями, разница в них только в поддерживаемых платформах и цене, хотя XMind сравнивался в пробной версии (бесплатная с ограниченным функционалом) (см. таблицу). В бесплатном и

открытом приложение VUE отсутствуют готовые шаблоны и структурный вид, но данные функции нужны далеко не всегда.

Таблица – Сравнительная таблица существующих информационных систем

ИС	XMind 2020	VUE	MindManager
Стоимость	Бесплатная неполная версия. Подписка 59.99$/год	Бесплатно	Триальная 30-ти дневная версия. Лицензия – 30 000 руб.
Платформы	Windows, MacOS, Linux, Android, iOS	Windows, MacOS, Linux	Windows, MacOS
Настройка тем	+	+	+
Экспорт изображения карты	+	+	+
Готовые шаблоны	+	-	+
Структурный вид карты	+	-	+
Поиск по карте	+	+	+
Свободное позиционирование	+	+	+

На основе анализа некоторых существующих систем видно, что явного функционала для построения когнитивных карт нет. Общий функционал узлов и связей, который есть во всех рассмотренных системах, позволяет построить граф для когнитивных карт.

В рамках выпускной квалификационной работы была спроектирована, разработана и задокументирована информационная система, специализированная на построении когнитивных карт. Для этого была произведена следующая работа:

1. Сделан обзор и выбор средств проектирования и реализации информационной системы:
   1.1. обзор UML инструментов [1];
   1.2. обзор кроссплатформенных фреймворков;
   1.3. обзор форматов для сохранения созданных карт.

На основе проведенного обследования было принято решение о выборе UML инструмента Draw.io, фреймворка Qt и формата JSON для сохранения когнитивных карт.

2. Спроектирована информационная система, удовлетворяющая следующим требованиям:
   a. предоставление пользователю возможности создания и изменения узлов, связей, веса связи;
   b. возможность свободного расположения узлов на поле;
   c. возможность поиска по слову или части слова в узлах и подсвечивания найденных узлов;
   d. сохранение созданной когнитивной карты в файл с форматом, понятном для разрабатываемой системы;
   e. возможность открыть сохраненную когнитивную карту из файла.
3. Разработан прототип интерфейса с помощью инструмента Qt Design Studio (см. рисунок)

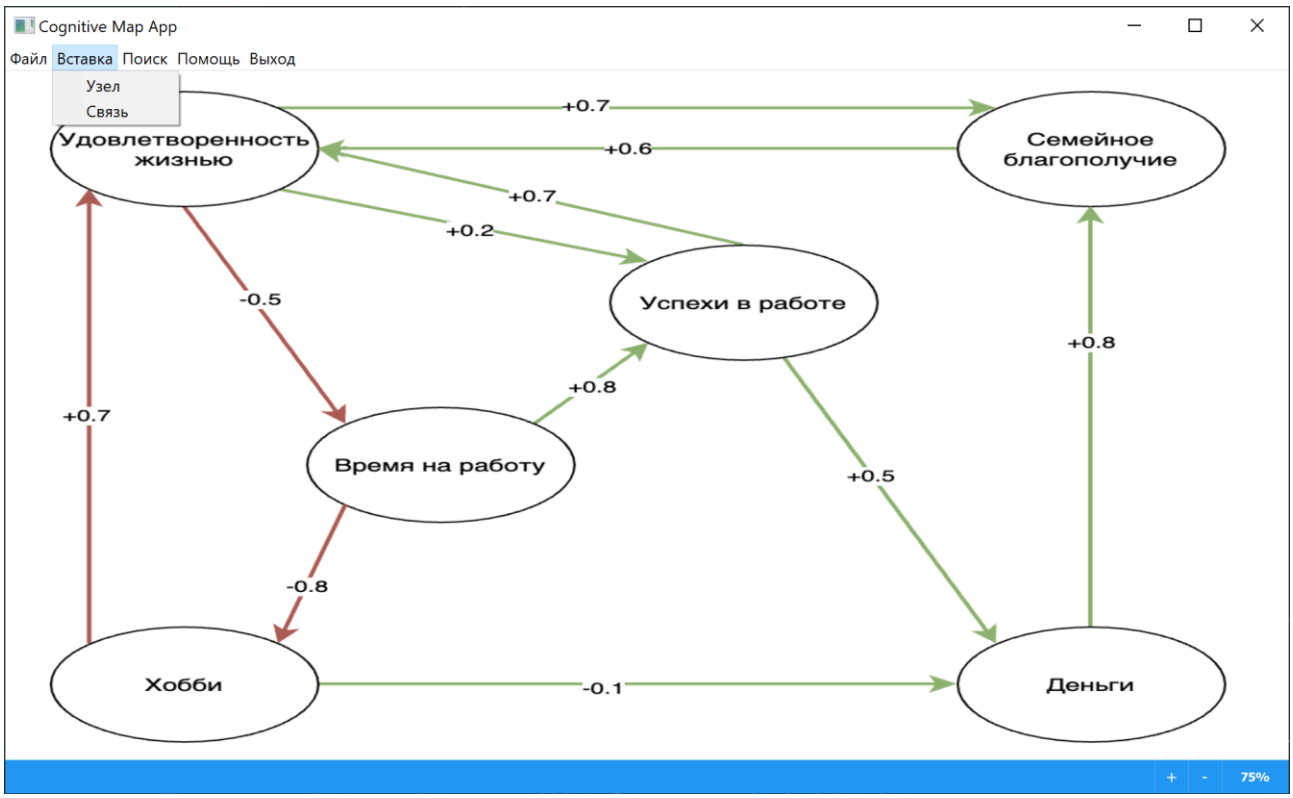


Рис. – Окно рабочей области с когнитивной картой

4. Описан процесс разработки информационной системы.

Для разработки любой современной информационной системы важно использовать систему контроля версии (далее – СКВ). СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать компонент системы, кто и когда сделал в коде какую-то ошибку, и многое другое. Использование СКВ также значит в целом, что, если потеряны файлы, спокойно можно всё исправить без особых дополнительных усилий.

5. Описаны компоненты приложения.

Используя язык QML, модули Qt Quick и Qt Quick Controls были выделены следующие компоненты интерфейса:

a. главное окно приложения;
b. стартовая страница;
c. страница с рабочей областью;
d. компонент Scene;
e. компонент Node;
f. компонент Relationship.

В результате выпускной квалификационной работы была успешно разработана информационная система «Когнитивные карты как инструмент формализованного описания предметной области», специализированная на построении когнитивных карт. За счет этапа проектирования – разработка потребовала минимальное количество времени.

Библиографический список

1. *Фаулер М.* UML. Основы, 3 е издание / М. Фаулер. – Пер. с англ. – СПб: Символ Плюс, 2004. – 192 с., ил.
2. MindManager. The World's Most Powerful Mind Mapping Software. [Электронный ресурс]. URL: https://www.mindjet.com/. (Дата обращения 19.04.2020).

3. Visual Understanding Environment. Flexible concept mapping tools for managing and integrating digital resources in support of teaching, learning and research. [Электронный ресурс]. URL: https://vue.tufts.edu/. (Дата обращения 16.04.2020).
4. XMind 2020. A full-featured Swiss Army Knife for your brain. A modern replacement for pencil and paper. [Электронный ресурс]. URL: https://www.xmind.net/xmind2020/. (Дата обращения 12.04.2020).

DESIGN, DEVELOPMENT AND DOCUMENTATION OF THE INFORMATION SYSTEM «COGNITIVE MAPS AS A TOOL FOR FORMALIZED DESCRIPTION OF THE SUBJECT AREA»

Zryachih Vladimir A.

Perm State University, 15, Bukireva st., Perm, 614990, Russia, amfik0159@gmail.com

This article discusses the definition of a cognitive map, a comparative analysis of existing information systems that have the necessary or similar functionality for building cognitive maps. The results are presented in a comparative table. Based on the analysis of systems, it can be seen that there is no explicit functional for building cognitive maps. The common functionality of nodes and links, which is in all considered systems, allows you to build a graph for cognitive maps. Therefore, a brief overview of the design tools was made, and the development process and components of the information system were described. The requirements that the developed system must satisfy are determined. During the design of the information system, models were obtained in the form of a use case diagram, a sequence diagram and a class diagram that satisfy the requirements for the system being developed. And also a prototype of the interface is presented.

Keywords: information system design, cognitive map.

Научное издание

Актуальные проблемы математики, механики и информатики 2021

Сборник статей по материалам студенческой конференции
(г. Пермь, 25 мая – 10 июня 2021 г.)

Ответственный редактор
А. П. Шкарапута

Издается в авторской редакции
Компьютерная верстка: *Т. С. Карина*

Объем данных 9,27 Мб
Подписано к использованию 08.07.2021

Размещено в открытом доступе
на сайте www.psu.ru
в разделе НАУКА / Электронные публикации
и в электронной мультимедийной библиотеке ELiS

Издательский центр
Пермского государственного
национального исследовательского университета
614990, г. Пермь, ул. Букирева, 15