Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЛЕСОТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ имени С.М.Кирова»

Кафедра информационных систем и технологий

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ: ТЕОРИЯ И ПРАКТИКА

Сборник научных трудов

Выпуск 9

Санкт-Петербург

2017

Рассмотрен и рекомендован к изданию
Научно-методическим советом
Санкт-Петербургского государственного лесотехнического университета

С о с т а в и т е л ь
М.А.Шубина, кандидат технических наук, доцент (отв. секретарь)

Р е ц е н з е н т
Доктор технических наук, профессор
Национального минирально-сырьевого университета «Горный»
И.В.Иванова

Информационные системы и технологии: теория и практика:
сб. научн.тр. вып. 9 / отв. ред. А.М.Заяц. – СПб.: СПбГЛТУ, 2017. – 158 с.
ISBN 978-5-9239-0779-7

Сборник подготовлен по материалам кафедры вуза, представленным на научно-технической конференции института леса и природопользования СПбГЛТУ в феврале 2017 г., и практических работ, выполненных ее сотрудниками.

Темплан 2017 г. Изд. №
ISBN 978-5-9239-0779-7

А. М. Заяц, кандидат технических наук, профессор

ОСНОВНЫЕ ИТОГИ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЫ И НИРС

1. Итоги научно-исследовательской работы

Профессорско-преподавательский состав кафедры продолжил проведение исследований фундаментального и прикладного характера в рамках госбюджетной научно – исследовательской работы «Информационные системы и технологии: теория и практика». Основные результаты были опубликованы в сборнике научных трудов кафедры № 8, который был подготовлен по материалам исследований и научно - практических работ сотрудников кафедры, представленных на научно-технической конференции института леса и природопользования СПбГЛТУ в январе 2016г. Все сборники научных трудов кафедры (1–8) зарегистрированы на платформе научной электронной библиотеки eLIBRARY.RU и имеют Российский индекс национального цитирования (РИНЦ).

Преподаватели кафедры приняли активное участие в **3** зарубежных, **7** международных и **11** российских научных конференциях.

По материалам исследований подготовлено более 40 научных статей, которые опубликованы в российских и зарубежных журналах. Все статьи имеют РИНЦ, 12 статей опубликованы в журналах из списка ВАК.

Оценка результативности и эффективности деятельности, публикационной активности и цитируемости ученых кафедры по индексам научного цитирования международных и российских баз данных следующая:

- индекс Хирша (Web of Science) - профессор Богатырев В.А. – **1**, доцент Жук Ю.А. – **2;**
- индекс Хирша (Scopus) - профессор Богатырев В.А.– **3**, доцент Жук Ю.А. – **3;**
- индекс Хирша (Elibrary - РИНЦ Science) - профессор Богатырев В.А – **22,** доцент Жук Ю.А. - **3**, доцент Васильев Н.П.- **2**, профессор Заяц А.М. – **1,** доцент Хабаров С.П. – **1.**

Продолжена практика создания прикладных программных продуктов, в этом году получены **3** свидетельства о государственной регистрации программ для ЭВМ. Авторы преподаватели кафедры: Васильев Н.П., Заяц А.М., Логачев А. А., магистранты и бакалавры. Отправлены на регистрацию еще **3** заявки.

Важным показателем в научной работе является деятельность преподавателей в области научно – методической работы, выражаю-

щаяся в написании учебных и методических пособий и статей. Так в 2016 году преподавателями кафедры подготовлены и изданы следующие учебные пособия:

- Богатырев В А. ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ. ТЕОРИЯ НАДЕЖНОСТИ. Рекомендовано УМО высшего образования в качестве учебного пособия для студентов вузов обучающихся по инженерно-техническим направлениям -Москва .:Юрайт, 2016. -318 с.
- Уткин Л.В., Жук Ю.А. Элементы системного анализа: теория игр 09.03.02-Санкт-Петербург .:Изд-во Политехн. ун-та, 2016. -84 с.
- Полетаева Н.Г. Основы построения распределенных информационных систем -СПб .:ИПО СПбГЛТУ, 2016. -128 с.
- Шубина М.А. Управление данными. Рекомендовано научно-методической комиссией СПбГЛТУ для направлений подготовки 09.03.02 и 09.04.02 .:СПбГЛТУ , 2016г.-132 с

Полностью подготовлены и находятся на завершающей стадии к изданию учебные пособия:

- Хабаров С.П. Шилкина М.Л. Вычислительные машины, системы и сети: Учебное пособие под ред. к.т.н., проф. Заяц А.М. Изд. Политехн. ун-та, 240 с.
- Шубина М.А. Информационные технологии. Учебное пособие для бак. 35.03.02. СПб.: СПбГЛТУ, 116 с.

Переработаны в соответствии с ФГОС 3+ более 30 рабочих программ учебных дисциплин по различным направлениям подготовки бакалавров и магистров. Разработан и рассмотрен на ученом совете университета учебный план направления подготовки «Лесное дело» по магистерской программе «Информационные системы и технологии в лесном хозяйстве»

2. Итоги научно-исследовательской работы студентов

На кафедре традиционно активно ведется научно-исследователь-ская работа студентами, которые являются активными членами СНО. Эта работа проводится по различным направлениям.

2.1. Участие студентов в олимпиадах, конкурсах и грантах

Три команды студентов 3 и 4 курсов успешно выступали в четвертьфи-нальных отборочных студенческих соревнованиях командного чемпионата мира по программированию - ACM ICPC 2016-2017, NEERC, Northern Subregional Contest (тренеры доц. Лебедев М.О., ст. преп. Курилова М.Н.).

Девять студентов 3 и 4 курсов приняли участие в Открытой международной студенческой интернет - олимпиаде (OIIO - OPEN INTERNATIONAL INTERNET-OLYMPIAD), по направлению подготовки «Информационные системы и техно-логии» и продемонстрировали хорошие знания (тренер ст. преп. Курилова М.Н.).

Студентка – выпускница 2016 года Калашникова Елена стала (второй год подряд) финалисткой конкурса «**Honor Cup**» (спонсор фирма « **Huawei**») международной олимпиады в сфере информационных технологий «**ИТ ПЛАНЕТА**» 2015/2016.

Студенты 4 курса Шалаев Е. И. и Васильев С. П. стали призерами конкурса компании СКБ Контур "Программирование : C#, ASP.NET MVC" и награждены дипломами за 3-е место.

В олимпиаде «**ИТ ПЛАНЕТА**» 2016/2017 участвуют и проходят тестовые испытания 19 студентов 1,2,3 и 4 курсов. Финал в мае г. Сочи (тренер проф. Заяц А.М.). Три студента института леса и природопользования СПбГЛТУ направления подготовки «Информационные системы и технологии» Раудсон А.А. - 3 к., Васильев С.П. - 4к., Шалаев Е.И. - 4к. успешно справились с тестом первого тура, набрав от 180 до 230 баллов из 300 и продолжат участие во втором туре.

Магистр Кылосов А. Ю. – победитель конкурса видеороликов в номинации «Моя будущая профессия», проводимого в университете под девизом «Это мой университет!». Видеоролик активно используется в профессионально - ориентационной работе приемной комиссии СПбГЛТУ.

Думов Максим Иннокентьевич, 4 курс, 7 гр. – победитель конкурса грантов 2016 года для студентов вузов, отраслевых и академических институтов, расположенных на территории Санкт-Петербурга (20000 руб.). Представлена исследовательская работа «Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей» (научный руководитель проф. А.М. Заяц).

На базе кафедры проведен региональный этап Всероссийской олимпиады профессионального мастерства 2016 по специальности 09.02.04 "Информационные системы". За успешную подготовку призера (студентка Михайлова О.П.) доцент кафедры Полетаева Н.Г. награждена специальным дипломом Комитета по науке и высшей школе Правительства Санкт - Петербурга.

Студенты 3 и 4 курсов института леса и природопользования СПбГЛТУ направления подготовки «Информационные системы и технологии» (специалист - руководитель магистр 2 курса Пушкарева Л.Г.) были включены в технический персонал приемной комиссии, для работы в качестве операторов – исследователей автоматизированной системы «Абитуриент», успешно справились с поставленными задачами и были поощрены руководством университета.

2.2. **Участие студентов в написании научных статей и выступлениях на научных конференциях**

Студенты являются авторами и соавторами 7 статей в сборниках докладов различных научных конференций и 6 свидетельств о государственной регистрации программ для ЭВМ. Наиболее значимыми, из которых являются**:**

• Пушкарева Л. Г. Применение современных информационных технологий при проектировании лесных питомников. Сборник трудов VII Научно-практической конференции молодых ученых «Вычислительные системы и сети - Майоровские чтения»/ Под ред. д.т.н., проф. Т.И. Алиева. СПб: Университет ИТМО, 2016.

•Пушкарева Л.Г. «Информатизация процессов проектирования и функционирования лесных питомников на основе реинжиниринга бизнес-процессов и внедрения сетевых технологий» Сборник научных трудов XIX Российской научной конференции «Инжиниринг предприятий и управление знаниями» (ИП&УЗ-2016).

•А.М.Заяц, Думов М.И. Обзор беспроводных сенсорных сетей и технологий информационных систем оценки лесной пожароопасности и мониторинга лесов // Информационные системы и технологии: теория и практика: сборник научных трудов- СПб..:СПбГЛТУ, 2016, №8.

•С.П.Хабаров, Е.И. Шалаев, С.П. Васильев. Реализация модуля логического вывода в составе информационной системы классификации растений. //Леса России: политика, промышленность, наука, образование /Материалы НТК.-СПб. .:СПбГЛТУ ,2016. № 2.

•Полетаева Н. Г., Туктарев Д.О. Защита данных криптографическими средствами СУБД Oracle//Информационные системы и технологии: теория и практика. Сборник научных трудов.- СПб.:ИПО СПбГЛТУ, 2016,№ 8.

2.3. Стипендии Президента РФ, Правительства РФ и Снкт – Петербурга

За активное участие в научно исследовательской работе и достигнутые результаты студенты члены СНО кафедры были отмечены стипендиями Президента РФ, Правительства РФ и КНВШ Правительства Санкт- Петербурга:

- Пушкарева Любовь Геннадьевна, стипендия Президента РФ магистр 2 курс. Приказ Минобрнауки №707 от 04.07.2016;
- Шалаев Егор Игоревич, 4 курс, 7 гр. Приказ Минобрнауки №707 от 04.07.2016;
- Боброва Анастасия Игоревна магистр 2 курс, Распоряжение №113 от 29.09.2016 «О назначении и выплате именных стипендий Правительства Санкт-Петербурга студентам образовательных учреждений высшего образования и среднего профессионального образования»;
- Думов Максим Иннокентьевич, 4 курс, 7 гр. Приказ ректора СПбГЛТУ №1734/ск от 07.09.2016.

На кафедре успешно функционирует информационная система. Силами сотрудников кафедры и студентов на производственных практиках

продолжены работы по ее модернизации в основу, которой положены современные идеи и технологии – виртуализации и облачных технологий.

Это позволит эффективнее использовать оборудование для проведения научных экспериментов, усилит практическую направленность выпускных квалификационных работ. Результатом модернизации также станет расширение спектра технологий доступа студентов к электронным ресурсам СПбГЛТУ и специализированным базам по проведению исследований в сети Интернет и эффективного применения современных информационных технологий в образовательном процессе для качественного проведения всех видов занятий.

А. М. Заяц кандидат технических наук, профессор
Н.А. Дмитриенко студент

ПОДХОД К ОРГАНИЗАЦИИ ПЕРЕДАЧИ КРИТИЧНЫХ ДАННЫХ ДАТЧИКОВ В ИНФОРМАЦИОННОЙ СИСТЕМЕ МОНИТОРИНГА ЛЕСНЫХ ТЕРРИТОРИЙ

Существует множество вариантов построения беспроводных сенсорных сетей (БСС) в информационных системах мониторинга лесных территорий [1-8]. При достаточно больших размерах территорий мониторинга, предлагается организовывать БСС как совокупность подсетей (кластеров), связанных координаторами и шлюзом взаимодействия: «сенсорная сеть – корпоративная сеть информационной системы».

При выявлении событий приводящих к возникновению лесного пожара, важным является своевременность передачи «критичных» данных фиксируемых датчиками сети на узел координатор и шлюз и далее конечным пользователям системы. Здесь и далее под «критичными» данными будем понимать измерения зафиксированные датчиками по уровню превышающие заранее установленный порог.

Для эффективной работы и получения актуальных результатов необходимо, необходимо чтобы доставка этих данных для принятия решения осуществлялась бы не только оперативно, но и с минимальными задержками при организации передачи.

Анализ показывает, передачу трафика, порождаемого собственно датчиками в беспроводной сенсорной сети можно осуществлять в трех режимах: периодическая отправка данных, отправка данных по запросу и управляемая событиями отправка данных [9].

Периодическая отправка данных представляет собой регулярную отправку информации собранную датчиком. При этом никакие внешние события, кроме выключения сенсорного узла, не могут повлиять на такую отправку данных.

Суть передачи данных по запросу состоит в том, что координатором (шлюзом) сенсорному узлу оправляются запросы, и он инициирует отправку измеренных данных.

При управляемой событиями отправке данных, предполагается, что они передаются по инициативе сенсорного узла, но только в том случае, если они есть в наличии и критично отличаются, от предыдущей передачи, либо данные о значениях измерений достигли каких-то предельных уровней. Например, температурный сенсорный узел отправляет данные только в случае, если измеряемая датчиком температура достигла или превысила заранее заданную величину.

Для рассматриваемой системы мониторинга приемлемыми являются второй и третий режимы, если считать, что отправка данных по запросу также происходит вследствие некого события, порождаемого в сервере системы или на рабочей станции пользователя в результате анализа сложившейся обстановки.

Необходимым условием при реализации этих режимов является то, что на время работы системы мониторинга, особенно в критических ситуациях, должны быть «открытыми» (в обе стороны) каналы передачи данных:

1. Конечные узлы беспроводной сенсорной сети (датчики) – координатор – шлюз.
2. Шлюз – веб-сервер – пользователи корпоративной информационной системы.

Из-за существенных различий в организации и протоколах передачи данных в сенсорной сети и за ее периметром реализацию этого условия предлагается решать по - разному для этих каналов.

Технически, объединение этих каналов осуществляется установкой шлюза за периметром БСС. Так как в каналах 1 и 2 используются различные среды передачи, а трафик между ними достаточно большой, то шлюз обычно реализован в виде отдельного устройства и может предоставлять точку доступа к сети Интернет. В состав устройства должен входить приёмопередатчик, совместимый с сенсорной сетью и с глобальной сетью типа GSM или WiMAX.

Анализ показывает [9], что можно использовать шлюз, имеющий аппаратно-программные средства для работы в сетях ZigBee и GSM, а также GPRS/EDGE канал для доступа к сети Internet. Предлагаемая реализация шлюза обеспечивают интеграцию и совместную работу сетей с различными протоколами и технологиями функционирования.

Шлюз обеспечивает подключение БСС к web - серверу системы, причем данные могут передаваться как для обработки на серверы, так и прямо конечным пользователям.

Канал 1 - «конечные узлы БСС – координатор – шлюз» может быть организован путем передачи данных по сети GSM по технологии GPRS (General Packet Radio Service — «пакетная радиосвязь общего пользования»). Эта технология позволяет координатору сенсорной сети использо-

вать возможности сотовой связи и производить обмен данными, как с другими координаторами, включенными в сеть, так и с корпоративными сетям и Интернет через установленный шлюз.

Главной особенностью такой технологии, нужной в критических ситуациях обстановки, является возможность постоянного подключения датчиков к сети и получение активного виртуального канала связи.

На время передачи данных сенсору предоставляется реальный (физический) радиоканал, который в остальное время используется для передачи данных от других датчиков сети. Максимально возможная скорость обмена данными с помощью технологии GPRS теоретически может достигать 170 Кбит/с., что вполне удовлетворяет требованиям передачи данных в рассматриваемой системе мониторинга лесных территорий.

Для ускорения обмена можно использовать EDGE (Enhanced Data for Global Evolution) цифровую технологию беспроводной передачи данных для мобильной связи, которая является надстройкой над 2G и 2.5G GPRS - сетями. Подключение в сети по EDGE примерно в 3 раза быстрее, чем по GPRS, максимальная скорость передачи данных может составлять 474 Кбит/с.

Доступ к радиоканалу в сети производится под управлением координатора, который подключается по соответствующему алгоритму, привязанному к сигналам маяков. В рассмотренном варианте соблюдается необходимая ориентированность на содержание данных (управляемая событиями отправка данных). Здесь необходимость обмена определяется информационной составляющей сообщений. Запрашивается только информация с тех узлов, данные на которых превышают установленные уровни (например, температура окружающей среды выше критической, при которой может возникнуть пожар) [9].

Среди протоколов, ориентированных на данные, следует отметить Sensor Protocols for Information via Negotiation (SPIN), Directed Diffusion protocol, Energy-Aware Data-Centric Routing (EAD) [10].

В рассмотренном варианте «ведущим» в обмене является координатор, принимающий информацию от датчика, зафиксировавшего скачок в измерениях, данные о котором немедленно передаются координатору, то есть происходит немедленная передача сообщений по открытому каналу 1. Полученная шлюзом информация от координаторов БСС подвергается обработке, где выявляются данные с критическими изменениями и далее они немедленно должны передаваться по каналу 2 на сервер для работы специальных клиент-серверных web – приложений. В результате исполнения, приложений на карте рабочего стола пользователя будет отображаться обстановка участка лесной территории в виде интерактивной графики и описательной информации необходимой специалистам лесного хозяйства для оценки обстановки и принятия решения on – line [11 -14].

При этом оперативность принятия решений будет зависеть также и от скорости выполнения web – приложений, которые должны исполняться так же быстро как, например, десктопные приложения.

В классической клиент-серверной Web - среде для получения данных и их использования приложениями клиент должен сделать HTTP – запрос и потом получать данные по известной схеме: установление TCP – соединения, получение «порции» данных от сервера и разрыв соединения.

В критических ситуациях, такая схема не может обеспечить минимальную латентность, так как инициатором обмена всегда является клиент (в нашем случае - шлюз), а данные обрабатываются на сервере, но он в этой схеме лишь источник данных, а не инициатор сетевого обмена.

Для устранения этого недостатка предлагается использовать технологию организации обмена по протоколу WebSockets.

Реализация этого протокола обеспечивает механизм быстрого и безопасного двустороннего информационного взаимодействия между клиентом и сервером при этом инициаторами обмена являются обе стороны. Заголовочный файл для установления соединения с сервером отправляется только один раз, все последующие обмены осуществляются без отсылки заголовка, что положительно влияет на трафик обмена и латентность передачи.

Подробно протокол Websocket рассматривается в [15-18], где представлен материал, по данной технологии и описываются возможные варианты ее применения в различных областях.

При использовании этой технологии сравнительно легко разрабатывать сложные одностраничные web-приложения с множеством различных асинхронных элементов на странице, которые обычно характерны для информационных систем мониторинга лесных территорий.

В рассмотренном подходе передачи критичных данных представлены основные идеи его применения, требующие детальной проработки во всех аспектах его аппаратно - программной реализации. Авторы лишь попытались показать возможность объединения и наиболее перспективных технологий обмена и их совокупного использования в гетерогенных web – ориентированных информационных системах мониторинга лесных территорий.

Библиографический список

1. А. М. Заяц, А.А. Логачев. Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей //Известия Санкт-Петербургской лесотехнической академии-СПб.: СПбГЛТУ, 2016. № 216, с. 241-255.

2. А. М. Заяц. Беспроводные сенсорные сети в системе мониторинга состояния лесов//сборник докладов международной конференции «Леса России» - СПб. .:СПбГЛТУ, 2016. № 2, с. 23-26

3. А.М.Заяц, М.И. Думов Обзор беспроводных сенсорных сетей и технологий информационных систем оценки лесной пожароопасности и мониторинга лесов//Информационные системы и технологии: теория и практика: сборник научных трудов - СПб.: СПбГЛТУ, 2016. № 8, с. 5-9.

4. Б. С. Гольдштейн Б. С. Сети связи пост NGN / Б. С. Гольдштейн, А. Е. Кучерявый. — СПб.: БХВ Петербург, 2014. —160 с.: ил.

5. П.А. Калантаев, В.П. Пяткин. Web – семантическая сенсорная сеть мониторинга окружающей среды. ИВМиМГ СО РАН, Новосибирск.

6. Богатырев А.В., Богатырев В.А. Надежность функционирования кластерных систем реального времени с фрагментацией и резервированным обслуживанием запросов // Информационные технологии - 2016. - Т. 22. - № 6. - С. 409-416.

7. Богатырев, В. А. Информационные системы и технологии. Теория надежности: учебное пособие. — М. : Издательство Юрайт, 2016. — 318.

8. Турбин С.С. Выбор средств имитационного моделирования вариантов орга-низации резервированной коммуникационной системы. //Информационные системы и технологии: теория и практика: сборник научных трудов № 9 - СПб.: СПбГЛТУ, 2017.

9. sut.ru›doci/nauka/avtoref/Vybornova_AI_diss.pdf А.И. Выборнова Исследование характеристик трафика в беспроводных сенсорных сетях.

10. Singh S/K/ Routing Routing Protocols in Wireless Sensor Networks –A Survey / S.K. Singh // International Journal of Computer Science & Engineering Survey (IJCSES). - November 2010. - Vol. 1. - N 2

11. А. М.Заяц, А. А. Логачев. Математические модели для поддержки принятия решений по предупреждению лесных пожаров при ограниченном объеме исходных данных. //Известия высших учебных заведений. Приборостроение-Санкт-Петербург .:Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, 2016. № 5, с. 342-347

12. С.П.Хабаров. Организация гетерогенных ЛВС с терминальным доступом между ее узлами. //Известия СПбЛТА-СПб. .:СПбГЛТУ , 2016. № 216, с. 267-280

13. А.А. Никифоров. Обработка материалов аэрофотосъемки, полученных с помощью беспилоьного летательного аппарата. //Информационные системы и технологии: теория и практика: сборник научных трудов № 9 - СПб.: СПбГЛТУ, 2017.

14. А.С. Бондаренко. Статистическая обработка материалов лесоводственных исследований. :учеб. пособие/ А.С. Бондаренко, А.В. Жигунов. – СПб: Изд-во Политех. ун-та,2016. – 125 с.

15. С.П. Хабаров. Взаимодействие узлов сети по протоколу Web-Socket. //Информационные системы и технологии: теория и практика: сборник научных трудов № 9 - СПб.: СПбГЛТУ, 2017.

16. С.П. Хабаров. Использование утилиты WebSocketD для удаленного выполнения программ. //Информационные системы и технологии: теория и практика: сборник
научных трудов № 9 - СПб.: СПбГЛТУ, 2017.

17. Л.Г. Пушкарева. Исследование протокола WebSocket в лабораторной работе на примере сетевого взаимодействия в гетерогенных систе-

мах «WINDOWS –UBUNTU». //Информационные системы и технологии: теория и практика: сборник научных трудов № 9 - СПб.: СПбГЛТУ, 2017.

18. Электронный ресурс. Е. Лисицкий. WebSockets. Российские интернет – конференции. 2010г. http://profyclub.ru/docs/63.

Н.П. Васильев, кандидат технических наук, доцент

ГИБРИДНЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРИЛОЖЕНИЙ ДЛЯ МОБИЛЬНЫХ ПЛАТФОРМ

Введение

В последнее время наблюдается рост парка различных мобильных устройств, их вычислительной мощности и спроса со стороны пользователей. В связи с этим растет также интерес специалистов к разработке и изучению технологий создания приложений для мобильных платформ (операционных систем): Android, iOS, Windows Phone (WP), BlackBerry и др. Наиболее популярными и востребованными являются первые две платформы.

По указанным причинам в настоящее время практически все популярные среды разработки программ (SDK – Sofware Development Kit) поддерживают разработку для тех или иных мобильных операционных систем. При этом особое положение имеют так называемые «родные» - native среды разработки, которые поставляются непосредственно «владельцами» мобильных операционных систем. Это Android NDK (Native Development Kit) для Android, Xcode для iOS. Все среды в свободном доступе для скачивания.

Особое положение занимает разработка приложений под iOS. Дело в том, что Xcode можно развернуть только на компьютере от компании Apple и собрать мобильное приложение можно только в рамках этой среды. Особую сложность также представляет развертывание (deploy) и распространение (distribute) на реальных устройствах: iPhone и iPad. По этой причине в настоящей статье речь пойдет в основном об этой платформе, а также о механизмах защиты приложений от фальсификации и проверки их полномочий, не смотря на то, что эти механизмы универсальны и используются для любых приложений (разработанных с помощью гибридных или любых других технологий).

Если разработка мобильного приложения ведется в рамках native-среды на языке, который предусмотрен в этой среде, то такую разработку можно считать «родной», native разработкой. Для программирования в Xcode используется язык программирования objective-C и swift, для Android NDK – язык C++ (есть еще и Android SDK, где используется язык Java). В любом другом случае, если используются иные языки программирования, или SDK сторонних разработчиков, то разработка в той или иной мере является гибридной.

Например, если используется язык C#, то можно воспользоваться средой разработки xamarin (www.xamarin.com). Если используются языки HTML5/CSS и JavaScript, то следует использовать технологию cordova/phonegap, или фреймворк Appcelerator Platform (www.appcelerator.com). Конечно, термин «гибридный» в полной мере подходит именно для таких случаев. Еще раз следует подчеркнуть, что в настоящее время существует достаточно большое количество всевозможных технологий, фреймворков и сред программирования, позволяющих вести разработку приложений для мобильных платформ, обзор которых является отдельной темой. Большинство сторонних поставщиков тех или иных технологий придерживаются определенной ценовой политики, большинство их инструментов являются платными. Cordova/phonegap – это бесплатные продукты.

Следует отметить, что многие популярные JavaScript фреймворки хорошо интегрируются с базовыми технологиями cordova или phonegap. Это в полной мере относится, например, к фреймворку Sencha ExtJS.

В статье использовались материалы оригинальных руководств, доступных на портале Apple [1,2].

Cordova/PhoneGap

Phonegap - это одна из самых первых технологий разработки гибридных мобильных приложений, на основе которой работают многие другие фреймворки. Phonegap был создан примерно в 2008 - 2009 году внутри канадского проекта Nitobi, как среда с открытым исходным кодом, которая позволяла получить доступ к нативным функциям устройства из встроенного webview. Целью проекта было обеспечить возможность построения мобильных приложений исключительно на web-технологиях (HTML5/CSS и JavaScript), но с возможностью вызова нативного кода.

В 2011 году Adobe приобрела Nitobi и все права на Phonegap (www.phonegap.com). Исходный код ядра был передан Apache. Этот исходный код остался открытым, но ему понадобилось новое имя. После пары неудачных попыток, было выбрано имя Cordova – по названию улицы, на которой был расположен офис Nitobi (cordova.apache.org).

Таким образом, cordova и phonegap в настоящее время выступают как две разные платформы, хотя и связанные друг с другом: phonegap опирается на cordova. Обе позволяют разрабатывать приложения для различных мобильных операционных систем, фактически, используя один и тот же исходный код (это справедливо до определенной степени).

Их основные отличия, достоинства и недостатки:

- Cordova ориентирована исключительно на CLI (Command Line Interface), то есть - на командную строку. В то же время phonegap предлагает довольно простое настольное приложение (phonegap desktop app), более привычное для пользователей Windows, в рамках которого можно выбрать основные настройки и создать проект (при этом можно работать и с CLI).
- С помощью cordova можно выполнить сборку проекта под ту или иную мобильную платформу, протестировать приложение с помощью

эмуляторов и развернуть на реальном устройстве. Это справедливо с одной существенной оговоркой: на компьютере должна быть установлена нативная среда разработки. Таким образом, сборка, тестирование и развертывание фактически осуществляется с помощью «родных» средств разработки под ту или иную мобильную платформу.

- Сборка проекта phonegap под ту или иную мобильную платформу осуществляется в облачном сервисе. Для этого надо зарегистрироваться на сайте build.phonegap.com, подготовить проект для сборки, загрузить проект и получить результат сборки. Очевидное достоинство такого подхода – не требуется устанавливать native SDK (что особенно актуально для iOS сборки, которая иначе невозможна без компьютера от Apple). Предварительное тестирование проекта осуществляется с помощью специального бесплатного мобильного приложения (phonegap developer app). Предусмотрены варианты для любой операционной системы. Это фактически эмулятор, который непосредственно развернут на мобильном устройстве.
- Процесс сборки, тестирования, и развертывания на мобильном устройстве значительно усложняется для iOS. Фактически, эти процессы невозможны без Apple mac.

Формально Cordova - это модуль (пакет) популярной серверной платформы node.js (http://nodejs.org), построенной на основе движка от Google (известного как Google Chrome JavaScript Engine V8). Это бесплатный программный продукт, который можно скачать с указанного выше сайта и установить практически в любой операционной системе. Установка собственно Cordova ничем не отличается от установки обычного модуля node.js - надо воспользоваться менеджером пакетов npm (входит в дистрибутив node.js): npm install -g cordova.

PhoneGap это также пакет node.js и устанавливается аналогично: npm install -g phonegap. Этот модуль опирается на модуль Cordova, но может устанавливаться и использоваться отдельно.

Центральное положение в архитектуре приложения Cordova занимает WebView. Это фактически web-обозреватель, который и будет воспроизводить HTML5/CSS и JavaScript-код. Это native-компонент, который автоматически подгружается Cordova в окончательную сборку под ту или иную мобильную платформу. Конечно, этот обозреватель имеет ряд существенных отличий от обычного web-обозревателя. Например, поддержка специфичных для реального мобильного устройства событий:

deviceready – это событие возникает после загрузки как DOM, так и собственно самого нативного кода приложения Cordova;

pause – это событие возникает, когда мобильная платформа переключается на другое приложение;

resume – это событие возникает, когда операционная система вновь возвращается к исполнению приложения;

и другие, среди которых есть специфичные для той или иной платформы, например, для iOS или Android.

Архитектура приложения Cordova

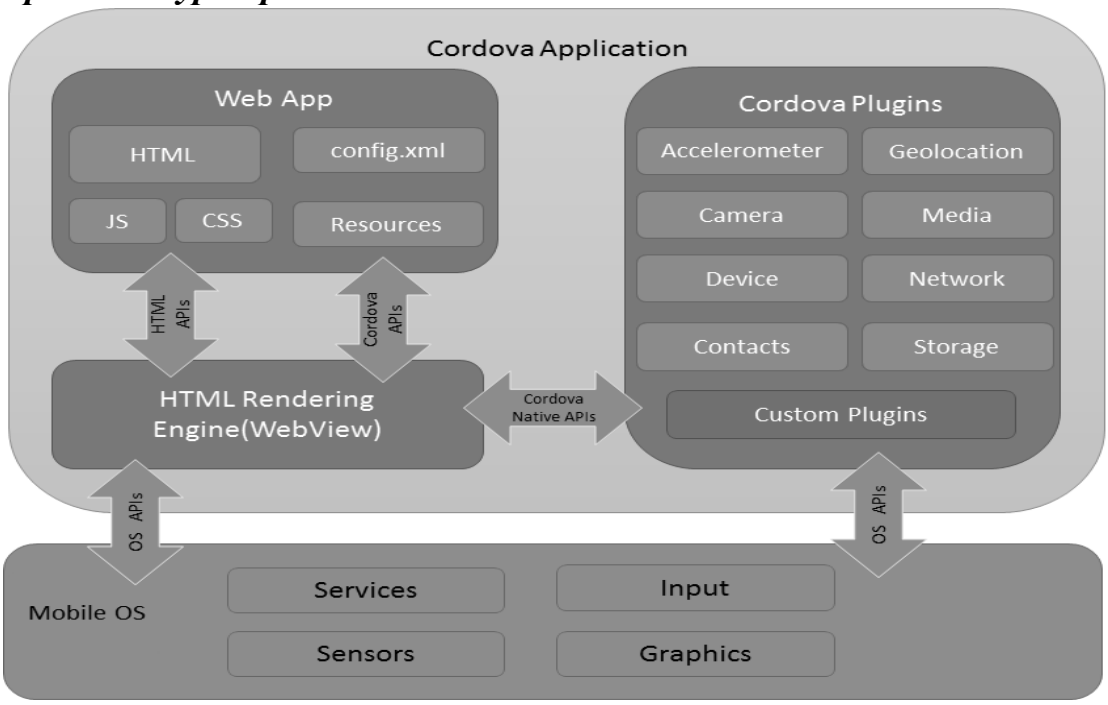


Рис. 1. Архитектура мобильного приложения, построенного на основе технологии Cordova

Кроме этого, WebView поддерживает API для связи с различными расширениями (плагинами) (например, для доступа к тем или иным дополнительным возможностям мобильного устройства: геолокации, акселерометру, камере и др.). Это, по сути, возможность исполнения нативного кода из JavaScript, выполняющегося в рамках WebView. Различают «Core Plugins», разработанные в рамках проекта Cordova, и плагины от сторонних разработчиков.

Наконец, само Web-приложение, разработанное на языках HTML5/CSS и JavaScript. Конечно, это приложение должно учитывать небольшое разрешение мобильных устройств. В настоящее время существует достаточно большой выбор фреймворков на JavaScript, которые ориентированы на мобильные устройства. Это – jquery mobile, sencha extjs modern, sencha touch и множество других.

Установка iOS приложений (deploy and distribute)

Процесс развертывания, то есть загрузка и установка (deploing) на мобильное устройство iOS приложений, тем более, их централизованное распространение (distributing) – довольно сложный и не бесплатный процесс. Следует, однако, подчеркнуть, что с появлением Xcode 7-ой версии стало возможна установка приложений без участия в одной из платных программ Apple (Apple Developer Program или Apple Developer Enterprise Program), то есть бесплатно, правда, с рядом существенных ограничений. Эту возможность иногда называют Free Provisioning. Пожалуй, самое существенное ограничение состоит в том, что приложение будет запускаться на мобильном устройстве только в течение недели. Однако если приложе-

ние вновь развернуть на мобильном устройстве, то оно снова просуществует очередную неделю.

В остальных случаях потребуется приобщиться (зачислиться - Enroll) к Apple Developer Program или Apple Developer Enterprise Program на портале developer.apple.com, а процесс разработки будет выглядеть следующим образом.

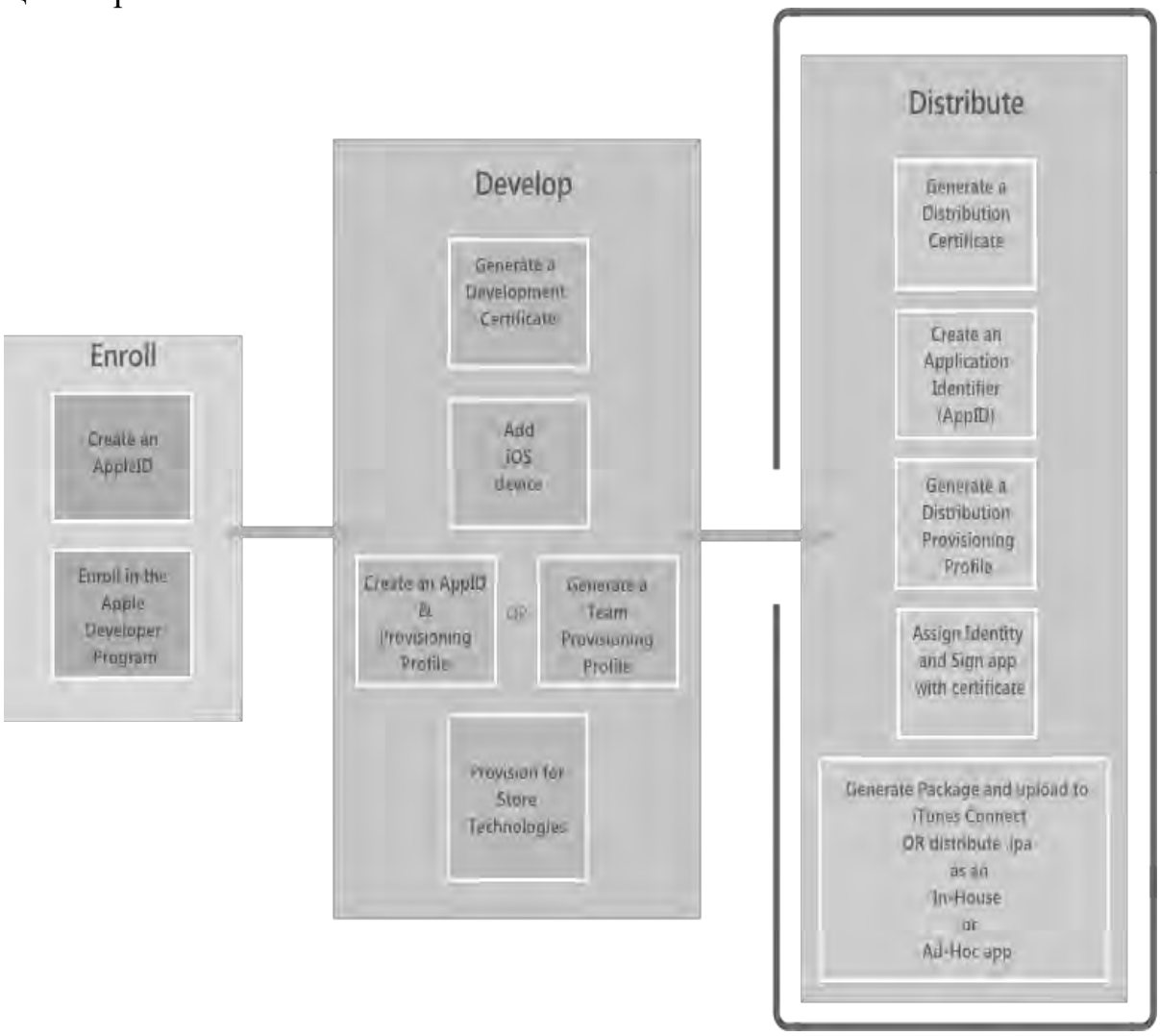


Рис. 2. Этапы разработки приложения под iOS

Различие этих двух программ в основном состоит в том, что участие в первой дает право на централизованное распространение приложений через App Store. Участие во второй ориентировано на распространение приложений только внутри фирмы (организации). Это так называемое In-House распространение (за пределами App Store). Участие в первой программе обойдется в 99$, участие во второй – 299$ в год. В рамках любой программы допускается так называемое Ad Hoc распространение (можно устанавливать приложения на 100 устройствах в год).

Возможности приложений, в основном, связанные с использование различных сервисов Apple, для различных программ будут различаться. Эти различия представлены в таблице (колонки iOS и iOS Enterprise).

Capability	iOS	iOS Enterprise	watchOS Extension	watchOS App	tvOS	Mac	Mac Developer ID
App Groups	●	●	●	●	●	●	○
Apple Pay	○		○				
App Sandbox						●	○
Associated Domains	○	●	○		○		
Background Modes	●	●		●	●		
Data Protection	●	●	●		●		
Game Center	○				○	○	
Game Controllers					●		
HealthKit	●	●	●				
HomeKit	●	●	●		●		
iCloud: CloudKit	○	●	○		○	○	○
iCloud: iCloud Documents	○	●	○			●	○
iCloud: Key-Value Storage	○	●	○		○	○	○
In-App Purchase	○				○	○	
Inter-App Audio	●	●					
Keychain Sharing	●	●	●		●	○	○
Maps	●	●	●		●	○	○
Personal VPN	○	●				○	○
Push Notifications	○	●	○		○	○	○
Wallet	○	●	○				
Wireless Accessory Configuration	●	●					

Рис. 3. Все сервисы (Capability) требуют Apple ID, для светлых меток – дополнительно требуется участие в платной программе разработчиков Apple

Provisioning Profile

Механизм сборки, развертывания и запуска приложений на мобильном устройстве под управлением iOS, предусмотренный Apple, направлен на решение следующих основных задач:

- Обеспечение безопасности. Прежде всего, это предотвращение фальсификации, то есть подмены, или модификации кода приложения с момента его установки.

- Контроль полномочий приложения. Прежде всего, это контроль возможности выполнения приложения на данном устройстве. Здесь проверяется сертификат разработчика и допуск данного приложения для выполнения на данном устройстве (с имеющимся у него идентификатором UDID).

Этот механизм предусматривает необходимость установки в одной связке с приложением, с одной стороны, так называемого профиля обеспечения (Provisioning Profile). С другой стороны, требуется цифровая подпись приложения сертифицированным разработчиком.

В профиле указываются три основные позиции: кто, то есть разработчик, что, то есть приложение, и где, то есть устройства, на которых это приложение может выполняться. Эта информация проверяется при запуске приложения (сопоставляется с соответствующими метаданными, которые извлекаются непосредственно из самого приложения) и по результатам проверки приложение либо запускается, либо его запуск отклоняется.

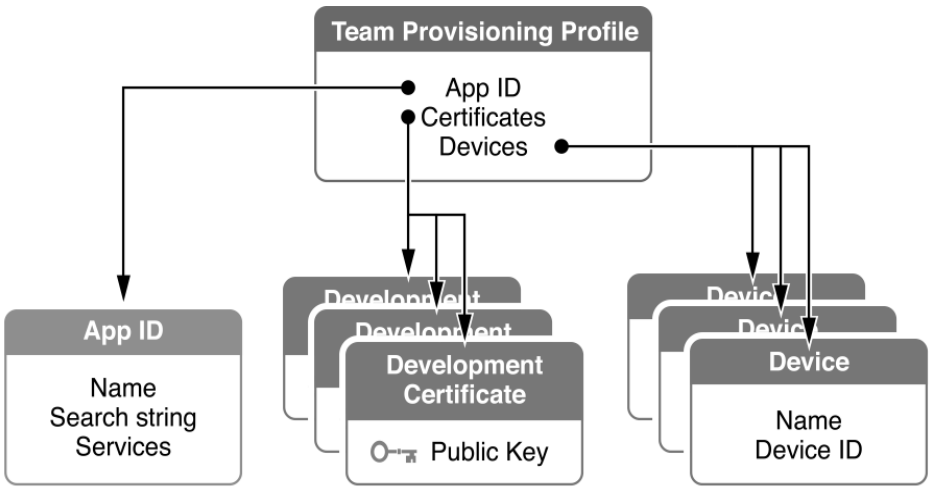


Рис. 4. Содержимое профиля обеспечения

Создание и модификация профиля обеспечения осуществляется опосредованно с помощью Xcode для команды разработчиков. В команде каждый член может играть определенную роль и быть агентом, администратором или простым пользователем. В зависимости от роли участник команды получает те или иные полномочия. Создатель команды является ее агентом и имеет максимальные полномочия.

Командный профиль обеспечения (Team Provisioning Profile) допускает подпись приложений любым членом команды и их запуск на всех устройствах членов команды. Команда может состоять также из одного члена – агента этой команды.

Следует подчеркнуть, если разработчик является участником платной программы, то Xcode будет работать совместно с порталом разработчиков (developer.apple.com) и запрашивать создание необходимых активов на этом портале. Активы (в том числе и профили) хранятся под учетной записью разработчика (Apple ID). Компания Apple также предусматривает непосредственную работу с порталом.

Xcode создает профиль обеспечения и его компоненты по мере необходимости. Прежде всего, Xcode запрашивает сертификат, если он отсутствует. Сертификат используется для идентификации разработчика. В профиле также должно быть зарегистрировано мобильное устройство (или устройства), на котором предполагается развернуть приложение, поэтому Xcode может попросить подключить это устройство. Xcode также выполняет требуемые настройки профиля для каждого сервиса Apple (capability), который предполагается использовать.

В профиле также хранится так называемый App ID, которому должен соответствовать Bundle ID приложения. Этот Bundle ID является уникальным идентификатором приложения (он, естественно, обязательно прописывается в коде сборки приложения).

Магазин App store и сервисы Apple используют Bundle ID для однозначной идентификации приложения. Поэтому, после того, как приложение будет доступно в App store, его Bundle ID не может быть изменен. Идентификатор Bundle ID образуется аналогично DNS, только в обратном порядке. Например, Xcode соединяет идентификатор компании, ее название, с названием приложения, в результате, может получиться что-то похожее на строку: com.example.mycompany.MyFirstApp. Однако, в отличие от доменных имен, идентификаторы Bundle ID чувствительны к регистру.

Специальный идентификатор App ID в профиле обеспечения используется для идентификации одного или нескольких приложений. Этот App ID может быть явным (explicit), тогда он в точности повторяет Bundle ID приложения, или являться шаблоном - wildcard App ID, тогда он соответствует множеству Bundle ID, построенных согласно этому шаблону. Xcode изначально создает wildcard App ID, и явный App ID создается только в случае необходимости. Например, если приложение использует сервис Apple, который требует указания явного App ID (например, облачные сервисы iCloud). Таким образом, один профиль может быть использован несколькими приложениями.

Цифровая подпись приложений

Для подписи приложений используется пара связанных ключей: открытый и закрытый (секретный) ключи (Public and Private keys), которые в документации Apple обычно называются: signing identity.

Эти ключи формируются Xcode в процессе запроса и получения сертификата разработчика с помощью портала (если разработчик - участник платной программы) или Xcode действует самостоятельно (в рамках Free Provisioning). Сертификат с открытым ключом хранится на портале под учетной записью разработчика. Сертификат с открытым и закрытым ключом (то есть - signing identity) хранится в виде связки ключей (key chain) на компьютере, где установлен Xcode.

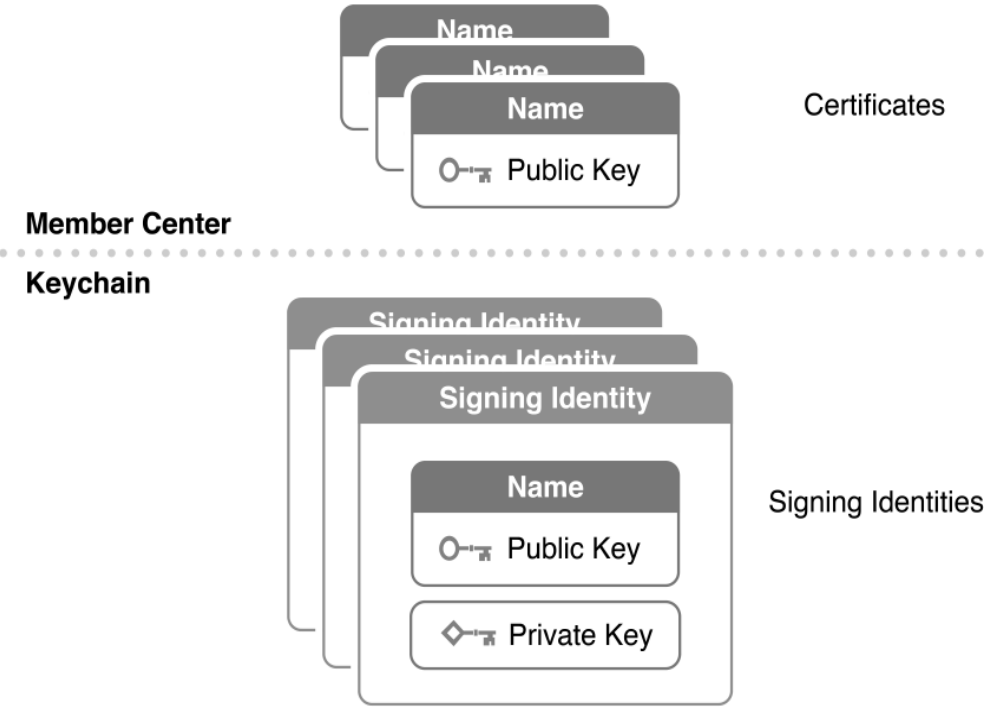


Рис. 5. Порядок хранения сертификатов

Важно отметить, если разработчик «переехал» на новый mac и не позаботился о сохранении сертификатов с секретными ключами, то он обеспечил себя дополнительной работой, связанной с пересозданием всех сертификатов и обеспечением новыми профилями всех своих приложений. В связи с этим в Xcode предусмотрена процедура экспорта и импорта сертификатов, которые необходимо хранить в надежном месте.

Под подписью кода понимают следующие три составляющие:

- Хеш (пломба, печать (Seal)). Это коллекция контрольных сумм или хэшей различных частей сборки приложения, созданных с помощью специального программного обеспечения. Пломба позволяет обнаружить изменения кода (случайные или внесенные вирусом) с достаточно большой вероятностью.
- Сигнатура. Пломба шифруется с помощью связки ключей, взятых из сертификата разработчика. Это гарантирует целостность хеша.
- Правила, регламентирующие порядок проверки сигнатуры. Некоторые из них присущи проверяющему программному обеспечению (в зависимости от его целей). Другие определяются подписчиком и опечатаны с остальной частью кода.

Механизм подписи кода создает хеши - пломбу, обработав различные части сборки приложения (библиотеки, исполняемый код, ресурсы, файл Info.plist и так далее) согласно одностороннему алгоритму хэширования. На выходе получается серия хэшей или контрольных сумм, которые являются уникальными, но которые не могут быть использованы для восстановления исходной сборки приложения.

Проверяющий отрабатывает тот же алгоритм хэширования и сравнивает полученные результаты с оригиналом – хранящимся хешем. Несовпа-

дение указывает на фальсификацию приложения. Очевидно, подобная проверка действует только, если вредоносное программное обеспечение не подменило хранимый хэш. Шифровка хеша - сигнатура гарантирует невозможность такой подмены.

Шифрование хеша выполняется с помощью закрытого ключа, который доступен в сертификате разработчика - signing identity на этапе сборки приложения с помощью Xcode на mac компьютере разработчика. Дешифрование, необходимое на стадии проверки для сопоставления полученного и хранимого хешей, выполняется с помощью открытого ключа, хранимого в Provisioning Profile.

Библиографический список

1. https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html
2. https://developer.apple.com/library/content/documentation/Security/Conceptual/CodeSigningGuide/Introduction/Introduction.html

М.Р. Вагизов ассистент

РАЗРАБОТКА ИНТЕРАКТИВНЫХ ГЕОИНФОРМАЦИОННЫХ СИСТЕМ: ПРИНЦИПЫ ПОСТРОЕНИЯ И КОНСТРУИРОВАНИЯ СИСТЕМЫ

Для планомерного развития лесной отрасли разработка и использование геоинформационных систем на новых принципах взаимодействия пользователя с системой является актуальной задачей. Способы и методы проектирования отраслевых ГИС, решающих определённый круг задач, базируются в первую очередь на методологию построения общей структуры конструирования информационных систем с использованием компьютерных технологий. Одной из целей при разработке геоинформационных систем, является создание наиболее удобного и продуманного человеко-машинного интерфейса. Степень взаимодействия пользователя с системой может быть оценена через понятие – интерактивность.

В чём состоит отличие стандартизированного интерфейса от интерактивного? При решении целого ряда задач, в современных геоинформационных системах, необходимо проходить подготовку, в части определения функциональных возможностей работы системы. В частности, в программах, которые используют базы данных, требуется знание языка построения запросов (SQL) для правильного выполнения обращения к данным, что увеличивает время доступа к нужным данным. Решение этой проблемы лежит в создании готовых запросов в виде реализуемых функций. Дополнительной сложностью является навигационная скованность

функционала интерфейса системы, некоторые используемые параметры труднодоступны и могут не использоваться пользователями вовсе. Результат не продуманного интерфейса - длительная предподготовка и освоения всех возможностей ГИС, при этом, может потребоваться прохождение специализированных курсов обучения, изучение большого объёма сопровождающих материалов.

Наличие сложно-модифицированных для пользователей интерфейсов приводит к тому, что объём заявленных функций системы не используется в полном объёме. Более того, с психологической точки зрения, отсутствие интуитивно-понятных модулей в составе интерфейса системы программы, может являться причиной неспособности обработки больших объёмов информации мозгом, что сказывается на уменьшении времени работы с программной средой.

Современный, высокий уровень производительности программно-аппаратного обеспечения, должен вывести используемые на сегодняшний день геоинформационные системы на новый уровень взаимодействия с человеком, благодаря развитию именно интерактивных систем взаимодействия пользователя и системы.

Таким образом, на передний план научных технологических решений становится определение и разработка методов построения интерактивных геоинформационных систем. Цель состоит во внедрении стандартизированного интерфейса таких элементов, которые обладают готовым решением определённого спектра задач [1,2]. Для начала, необходимо определить, какими средствами и за счёт чего, будет достигаться необходимая интерактивность геоинформационной системы в зависимости от целей её использования. При этом вне зависимости от прикладных задач решаемых геоинформационной программой, можно выделить некоторые критерии, обязательное использование которых привёдет к улучшению взаимодействия с системой [3]:

1. модульное программирование;
2. продуманная архитектура информационной системы;
3. принцип декомпозиции при условии связности системы;
4. требования интуитивного взаимодействия с системой.

Необходимо, что бы геоинформационная система содержала в себе такие технологические решения, что бы обеспечить минимальное количество времени на решение главных задач и упростить рутинный поиск данных.

Первый аспект, на который необходимо обратить внимание, это применение критериев для улучшения взаимодействия. Большое внимание при решении вышеперечисленных критериев отражается на стадии проектирования информационной системы при её разработке. Процедура полного конструирования системы должна отражать её цели, функциональную реализацию, быть последовательна, логична и понятна не только проектировщику.

Второй аспект в создании интерактивного интерфейса - это интеллектуализация процедур поиска, навигации, управления и вывода результатов работы. Интерактивная система должна содержать:

1. модули в виде готовых решений (содержащих в себе функции);
2. готовые запросы без необходимости сложно-конфигурируемого ручного ввода;
3. качественное визуальное сопровождение;
4. доступную поддержку при возникновении сложностей работы с ней.

Следует отметить, что задача проектирования интерактивного интерфейса несёт в себе грамотную классификацию процессов системы, выявление необходимых функций по количеству обращений и итоговый доступный вид [4,5].

На рис.1 представлены основные компоненты и взаимосвязь системы с пользователем. Разберем уровень взаимодействия, определяющий интерактивную систему, как таковую. Наличие в системе готовых решений в виде блоков, часто используемых функций повышает скорость принимаемых решений. Лицо принимающее решение в данном случае, руководствуется только особенностью блока, подходит ли он для решения необходимой задачи или нет. Следующий модуль - интуитивная навигация. На стадии проектирования и конструирования системы, все выполняемые функции должны быть классифицированы не более чем на три перехода до её месторасположения. Более того, каждая программная система должна быть сконфигурирована, учитывая её направленность, и строиться из принципов и спецификаций предметной области. На сегодняшний день ситуация видится обратной, так при построении как web-сервисов, так и приложений используются шаблоны готовых решений, лишь слегка модифицируя и дополняя систему необходимыми модулями. Нельзя не отметить положительные стороны проектирования такого подхода, заключающегося, прежде всего в сокращении времени построения системы, а набор готовых библиотек так же существенно упрощает конфигурацию и структуру программы. Однако, отличительной особенностью интерактивных систем от шаблонных, является их качественная реализация, быстрое выполнение заявленных функций, готовые решения, уникальный стиль. Такие системы будут запоминаться в использовании, их функции могут быть быстро освоены, они приятны в эксплуатации с точки зрения психологического восприятия. Приведем наглядный пример реализации двух интерфейсов геоинформационных программ (рис. 2).

Отметим функции, усложняющие работу со стандартизированным интерфейсом:

1) отсутствие объединения в блоки;
2) реализация основных задач в виде списка;
3) большое количество переходов до момента выбора необходимой функции.
4) стандартное графическое оформление.

Приведем пример переделанного стандартизированного интерфейса в интерактивный вид (рис.3).

Рис. 1. Элементы интерактивной системы

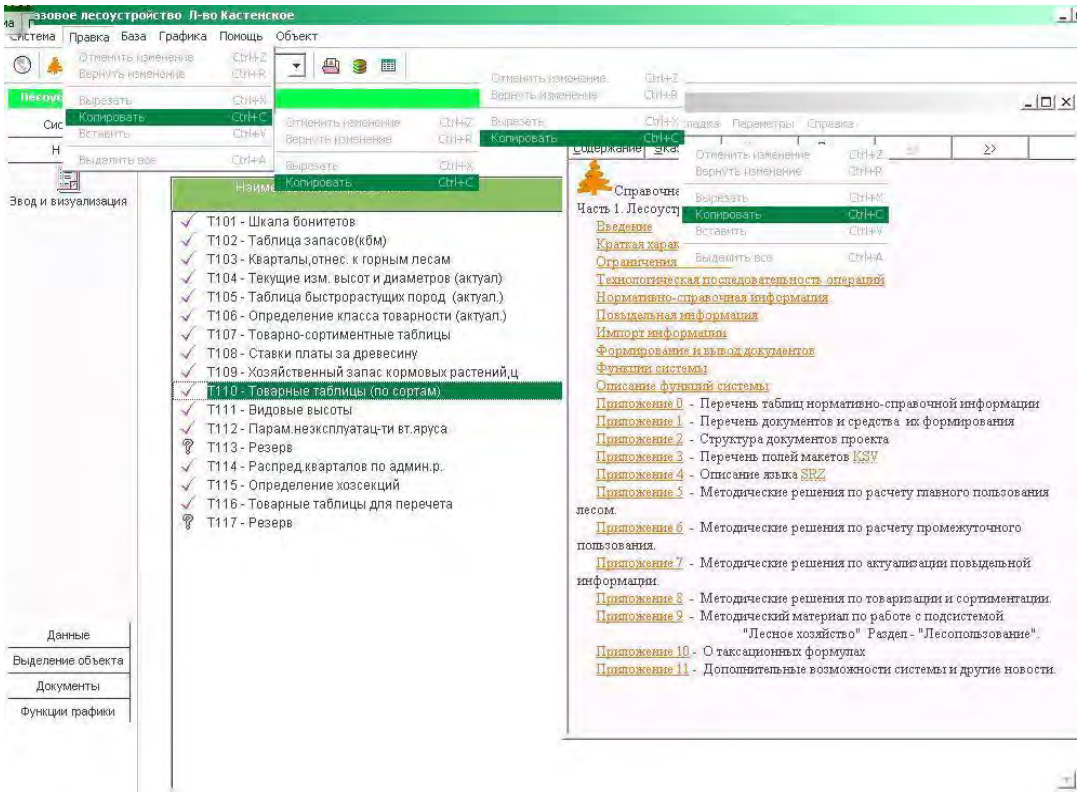


Рис.2. Геоинформационная программа с стандартизированным интерфейсом

Особенности изменения стандартизированного интерфейса:

1) основные задачи упорядочены в модули (блоки);
2) диалоговое окно текущей команды;
3) дополнительное меню часто используемых функций;
4) отсутствие «длинных» переходов;
5) наличие истории действий в системе.

Качественная реализация интерактивного интерфейса призвана: а) сокращать количество итераций для поиска и выполнения необходимой задачи, б) уменьшить время освоения и навигации в геоинформационной программе, в) ускорить процесс человеко-машинного взаимодействия.

Возможно ли, оценить систему с точки зрения её интерактивности? Для этого требуется ввести понятие интерактивность как отношение, суммы измеряемых величин (*Vr),* тогда:

$$V_r = \frac{P_k}{n * T_x}$$

где *Pk* количество действий одной функции программы, определяемое через сумму количества операций на время их выполнения, *n* – число обращений к главным функциям программы *Tx* – время реализации действий.

$$Pk = S1+S2+S3+Sn,$$

где *S1, S2, Sn* – количество действий одной функции.

Рис.3. Геоинформационная программа с интерактивным интерфейсом

Дополнительной оценкой может являться введение специальной шкалы, где информационная система оценивается с точки зрения её интерактивности. Обозначим 6 (шесть) уровней интерактивности системы принимая максимальную оценку как наиболее высокий параметр.

1. Система не интерактивна (сложность в навигации, в управлении, требуется значительное время для её освоения, интерфейс сложно структурирован);

2. Система обладает отдельными модулями, расположение главных функций затруднено, требуется ручной ввод запросов;

3. В системе присутствуют элементы интерактивного взаимодействия, более половины функциональных возможностей программы требуют дополнительного освоения.

4. Интерфейс системы преимущественно состоит из модульной структуры, в него включены сконфигурированные запросы в виде готовых решений;

5. Интерфейс системы достаточно интерактивен, интуитивно понятен в него входят реализованные функции;

6. Система интерактивна, проста в навигации и управлении, требуется минимальное время для её освоения, интерфейс сконструирован логично и последовательно.

На сегодняшний день несомненным является всплеск в развитии информационных технологий. Повышение производительности программно-аппаратных средств позволяет вывести геоинформационные системы совершенно на новый уровень взаимодействия с конечным пользователем. Разработка именно интерактивных ГИС позволит в первую очередь сократить время на решение прикладных задач. Стоит отметить, что грамотное конструирование интерактивной системы, не означает её упрощение, логически ситуация видится обратной, тем не менее способы подходы и методы проектирования современных геоинформационных систем требуют дальнейшего развития, изучения и поиск путей их применения.

Библиографический список

1. А. М.Заяц, А. А. Логачев. Математические модели для поддержки принятия решений по предупреждению лесных пожаров при ограниченном объеме исходных данных. //Известия высших учебных заведений. Приборостроение-Санкт-Петербург .:Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, 2016. № 5, с. 342-347

2. А. М. Заяц, А.А. Логачев. Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей //Известия Санкт-Петербургской лесотехнической академии-СПб.: СПбГЛТУ, 2016. № 216, с. 241-255.

3. Пресняков В.А. Проектирование информационных систем. Администрирование MS SQL Server: методические указания.

4. Вагизов М.Р. Сиразетдинова Г.Р. Разработка интерактивного картографического сервиса: описание структуры и механизмов взаимодей-

ствия системы// Лесная таксация и лесоустройство. -2015. -№1(52). -С. 56-61.

5. Гаскаров Д.В. Интеллектуальные информационные системы. Учеб. Для вузов. –М.: Высш. Шк., 2003. -431с.

Ю.А .Жук кандидат педагогических наук, доцент
И. **В.Сытюг** магистр

РАСПОЗНАВАНИЕ АНОМАЛЬНОГО ПОВЕДЕНИЯ СТУДЕНТОВ НА ОСНОВЕ АЛГОРИТМОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Введение

В настоящее время все большее распространение получает анализ изображений человека для идентификации его личности или его состояния. Применение данного анализа быстро приобрело большую популярность в разных областях человеческой деятельности. Распознавание человека по изображению лица отличается тем, что данный способ можно применять дистанционно, тем самым, не доставляя неудобств для человека, а также при выявлении подозрительных личностей, не подвергая опасности других людей. Для анализа достаточно: персонального компьютера, соответствующего программное обеспечение и изображений людей. В ходе анализа изображений можно не только идентифицировать личность человека, но и узнать его эмоциональное состояние.

Одной из главных причин использования распознавания образов в повседневной жизни является активный рост объемов получаемой информации, которую необходимо качественно и быстро обработать. Стоит отметить, что подавляющее большинство научных работ в данной области имеет эмпирический характер. Для эмпирического метода исследования характерно непосредственное изучение исследуемого объекта в ходе наблюдения за ним, и не требует, в отличие от теоретического, выявления сущности явления. Эмпирический метод заключается в сборе и подготовке информации, после чего, производится тестирование алгоритма.

Важнейшей областью, в которых применяется распознавание образов, является обеспечение безопасности аэропортов, метро, мест скопления большого количества людей. Еще одной жизненно важной областью человеческой деятельности, в которой применяется распознавание образов, является медицина, но, к сожалению, может возникать субъективная ошибка припо ряду причин, в том числе из-за не квалифицированности специалиста. Компьютерное же зрение может существенно облегчить и расширить современные возможности, выявляя заболевания не только на видимой для врача стадии, но и на более ранней.

Распознавание образов также применяется в игровой индустрии, для распознавания автомобильных номеров, в распознавании текстов, в геоло-

гии и во многих других областях человеческой деятельности. Ведущие мировые компании сегодня строят на процессе распознавания образов самые различные продукты.

Одна из немногих отраслей, куда не столь активно внедряются современные технологии интеллектуального анализа данных – это педагогика. В современном образовании все больше набирает популярность тестирование, как вид аттестации. Это связано с большим обхватом тестируемой аудитории, а также с быстрой проверкой результатов. Современный человек все больше выполняет различные тестирования, ведь это очень удобный вид аттестации, но до сих пор контролирует этот процесс человек. Порой невозможно проследить за всеми участниками тестирования, что ведет к возникновению некорректного (аномального) поведения - к использованию неразрешенных источников информации. Для того, чтобы автоматизировать идентификацию аномального поведения студентов, предлагается использовать алгоритмы машинного обучения, которые последнее время стремительно завоевывают практически все направления деятельности человека.

Существует множество методов распознавания образов, но все они имеют общую схему процесса обработки изображения [1], эта схема представлена на рисунке 1.

Рис. 1.Общая схема процесса распознавания образа

Этап 1. На вход подается изображение, на котором присутствует некий объект. На данном этапе происходит распознавание очертаний образа, локализация объекта на изображении и вычисление координат расположения объекта. Координаты передаются на следующий этап.

Этап 2. После получения координат объекта на изображении, происходит его выравнивание. Это необходимо, так как в повседневной жизни объекты могут изменять свое положение, и, если не учитывать этот фактор, то погрешность распознавания будет очень высокой.

Этап 3. На данном этапе изображение объекта уже локализовано и выровнено, поэтом наступает черед извлечения характерных признаков, на основе которых будет происходить распознавание. При этом стоит учитывать, что увеличение признаков иногда приводит к незначительным изменениям результата, при этом поглощает большое количество вычислительных и временных ресурсов.на выходе данного этапа формируется вектор характерных признаков.

Этап 4. На базе характерных признаков производится распознавание образа, то есть определения к какой именно категории относится объект.

Одним из наиболее эффективных и распространенных способов представления и решения перечисленных выше задач являются *искус-*

ственные нейронные сети[2,3].Нейронные сети получили свое распространение не так давно, но, несмотря на это, они заинтересовали общественность. Поскольку принцип их работы основан на принципе работы человеческого мозга, который, в свою очередь, до сих пор является для ученых загадкой. Отличительной особенностью человеческого мозга является его способность делать выводы, базируясь на предыдущем опыте, что помогает строить логические связи и приходить к новым результатам, а порой и открытиям.

Существует множество разновидностей нейронный сетей[4], но наилучшие результаты в распознавании образов и эмоций показали CNN (сверточные нейронные сети), имеющие особенную архитектуру, которую предложил Ян Лекун [5]. Она показана на рисунке 2.

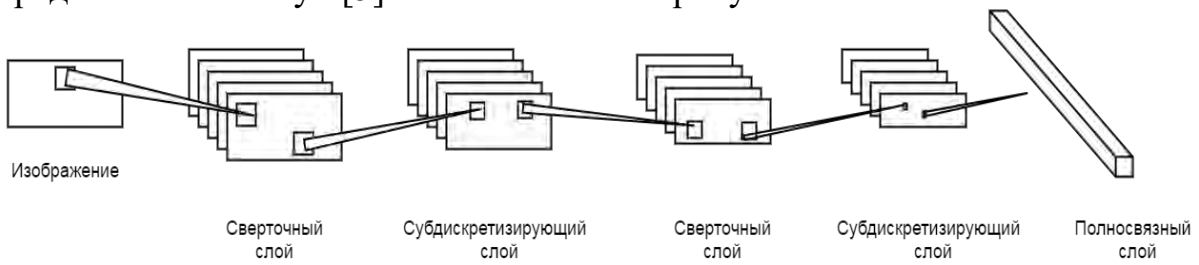


Рис. 2. Структура сверточной нейронной сети

Сверточная нейронная сеть построена по аналогии с работой зрительной коры мозга, которая в свою очередь состоит из простых и сложных клеток, простые - реагируют на прямые линии под разными углами, а сложные - связаны с активацией некоторого количества простых клеток[6].

Сверточные нейронные сети (СНС) оперируют тремя видами слоев: субдискретизирующими, сверточными и полноствязными. Основная идея СНС заключается в чередовании субдискретизирующих и сверточных слоев, причем первый слой - всегда сверточный. Чередование слоев необходимо для извлечения признаков при достаточно небольшом количестве тренируемых параметров. После прохождения слоев выходные данные называются картой параметров.

Субдискретизирующий слои предназначены для уменьшения размерности изображения, то есть заданное изображение уменьшается в несколько раз, что обеспечивает инвариантность к масштабу. Каждой карте слоя соответствует определенная область во входных данных, при этом ни одна карта не пересекается с другими во входных значениях. Изменения, происходящие с изображением после прохождения данного слоя, показаны на рисунке 3.

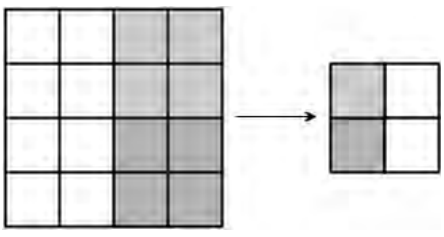


Рис. 3. Субдискретизирующий слой

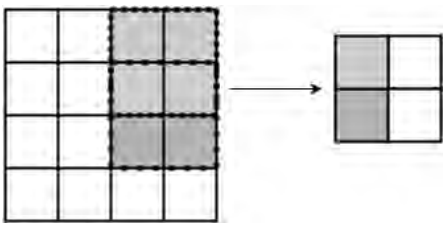


Рис. 4. Сверточный слой

Сверточные слои предназначены для уменьшения количества параметров в данном слое. В отличие от субдискретизирующего слоя, во входных данных изначальные данные карт расположены внахлест, как показано на рисунке 4.

Полносвязный слой - это набор выходных параметров, выделенных из исходного изображения.

Одной из начальных задач нашего исследования является составление баз данных, которые будут представлять из себя хранилище изображений студентов в матричной форме. Данные снимки будут получены в ходе прохождения тестирования учащемся с помощью веб-камер. Использование сверточных нейронных сетей для решения этой задачи обусловлено тем, что они обеспечивают частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. Второй этап исследования заключается в том, что будет проводиться анализ этих фото, а именно эмоциональное состояние человека, а также анализ положения головы, направление взгляда, всего, что может указать на аномально поведение учащегося в процессе тестирования. Для выявления эмоций человека, прежде всего, необходимо распознать изображение его лица, и правильно определить ключевые точки, что сейчас не является тривиальной задачей. После выделения ключевых признаков, будет проведено обучение нейронной сети. Для этих целей предлагается использование библиотеки Caffe, которая реализована с использованием языка программирования C++. Разработчики Caffe поддерживают возможности создания, обучения и тестирования полностью связанных и сверточных-нейросетей.

Разработка системы диагностирования аномального поведения студентов на основе интеллектуального анализа данных имеет широкий исследовательский потенциал. Полученные результаты могут быть применены в практике работы образовательных учреждений в качестве модуля в системы дистанционного обучения.

Библиографический список

1. Agrawal, D.D., Dubey, S.R. and Jalal, A.S. (2014) “Emotion recognition from facial expressions based on multi-level classification”, Int. J. Computational Vision and Robotics, Vol. 4, No. 4, pp.365–389.

2. Хайкин С., Нейронные сети полный курс. [Текст] / С. Хайкин. Москва: Вильямс, 2008. С. 1103.

3. WenYi Zhao. Imagebased Face Recognition: Issues and Methods [Электронныйресурс]. Электрон.текстовые дан. Режим доступа: http://www.face-rec.org/interesting-papers/General/Chapter_figure.pdf, свободный.

4. J.Ashok, Dr.E.G.Rajan. Principal Component Analysis Based Image Recognition [Электронныйресурс]. Электрон.текстовые дан. Режим до-

ступа: http://www.ijcsit.com/docs/vol1issue2/ijcsit2010010203.pdf, свободный.

5. Victor-Emil Neagoe, Andrei-PetruBărar, NicuSebe, Paul Robitu. A Deep Learning Approach for Subject Independent Emotion Recognition from Facial Expressions [Электронныйресурс]. Электрон.текстовые дан. Режим доступа: http://www.wseas.us/e-library/

6. Samer Hijazi, Rishi Kumar, and Chris Rowen. Using Convolutional Neural Networks for Image Recognition [Электронныйресурс]. Электрон.текстовые дан. Режим доступа: https://ip.cadence. com/uploads/901/ cnn_wp-pdf, свободный.

Ю.А. **Жук** кандидат педагогических наук, доцент
А.А. **Чигиринский** магистр

ОПРЕДЕЛЕНИЕ ФАКТОРОВ, ВЛИЯЮЩИХ НА УСПЕВАЕМОСТЬ СТУДЕНТОВ,С ПОМОЩЬЮ МЕТОДА ЛАССО

Учебная успеваемость всегда являлась одной из самых актуальных проблем образования. Развитие технологий, в том числе информационных, увеличение потока информации устанавливают новые, повышенные требования к профессиональным навыкам выпускников высших учебных заведений. Успех обучающегося, определенно, показатель,который зависит не только от оценок и индивидуальных результатов обучения в вузе, так и от жизненных условий и опыта полученного до поступления. Выявление наиболее значимых факторов, которые влияют на обучение, является не первый год актуальной задачей для десятков педагогов и психологов.

Система высшего образования располагает огромными объемами данных о студентах. Однако до сих пор факторы и конкретные причины, например, неуспеваемости, выявляются качественным анализом, без использования современных математических методов. Важнейшим инструментом реализации новых технологий являются эффективные алгоритмы и модели машинного обучения, которые являются необходимым элементом в технологической цепочке выявления значимых факторов, влияющих на успеваемость студентов и построение на основе интеллектуального анализа данных системдля принятия решений и прогнозирования результатов обучения.

Ряд исследований посвящён данной проблеме. В большинстве из них исследователи пытаются найти наиболее эффективный алгоритм для решения задач прогнозирования[3,4,5]. Проблемами исследований такого характера является, чаще всего, наличие большой обучающей выборки, данных с высокой степенью достоверности и качественной модели анализа этих данных, которая с большой точностью позволит отобрать действительно важные факторы влияния на успеваемость.

Современные методы интеллектуального анализа данных позволяют выполнять подобные прогнозы, учитывающие множество имеющихся факторов, используя классификационные или регрессионные модели анализа данных.Также существует задача определения, как меняется влияние на успеваемость различных факторов у студентов разных учебных направлений, а также у студентов различных курсов.

Целью работы является разработка модели и программного обеспечения для определения и ранжирования факторов, влияющихна успеваемость студентовразличных курсов.

Постановка задачи

Существует множество факторов, которые могут оказывать влияние на уровень успеваемости студентов, для удобства их можно разделить на группы[7]:

- психофизиологические и социальные факторы;
- организационные и экономические факторы.

К первой группе относятся личностные качества студентов, их психологические особенности. Изучая данные особенности, такие науки как педагогика и психология работают над совершенствованием существующих учебных методов и программ.

К личностным и психофизиологическим особенностям конкретно взятого студента можно отнести его интеллектуальные способности, уровень имеющейся подготовки, полученной до поступления в высшее учебное заведение, мотивацию к обучению, способность к организации самостоятельного обучения, различные интересы, черты характера. Подходящими для изучения факторами нужно также считать пол и возраст студентов, состояние их здоровья. Также для психологического состояния могут быть важны ощущение поддержки со стороны других людей, семьи, различная помощь.Воздействие на уровень успеваемости могут оказывать способности студента к коммуникации, установлению контактов с другими людьми (социальные факторы). Оценка своего собственного положения в обществе, умение выстроить общение с преподавателями.

Помимо личностных качеств, безусловно, необходимо принимать во вниманиеразнообразные организационно-экономические факторы. Нужно учитывать специфику организации обучения конкретного студента, форму обучения (очная, заочная, дистанционная и др.), уровень обучения, квалификацию преподавателей и еще множество факторов.

После отбора возможных факторов, возникает необходимость получения данных для возможности проведения их анализа. Во многих работают системы дистанционного обучения и информационные системы управления, из баз данных которых возможно получить информацию о студентах, и, определенным образом формализовав их, сопоставить с отобранными факторами.Если данных для исследования окажется недостаточно, то можно получить необходимые сведения путем анкетирования и опроса студентов.

Фактически, решение задачи выявления значимых факторов состоит в определении некоторой зависимости (функции) между студентами, представленными набором факторов, и некой шкалой, представляющей степень успеваемости.Выявление данной зависимости поможет определить, какие факторы более важны, а какие менее. Для решения данной задачи подходит использование методов и алгоритмов машинного обучения.

Выбор метода

Важными понятиями машинного обучения [2] являются объект, ответ, обучающая выборка, модель алгоритма.

Существует зависимость (неизвестная) отображения из множества объектов X во множество ответов Y:

$$y^*: X \to Y.$$

Существует также и конечное множество из nизвестных сопоставленных объектов и ответов, называемое обучающей выборкой:

$$X^n: \{(x_1,y_1), \dots ,(x_n,y_n)\}$$
$$\{x_1, \dots ,x_n\} \subset X ,\{y_1, \dots ,y_n\} \subset Y$$
$$y_i = y^*(x_i), i = 1 \dots n.$$

В контексте решаемой задачи, объектами являются студенты, ответами являются степени их успеваемости. Обучающая выборка представлена известными на текущий момент данными об обучающихся студентах и степенью их успеваемости.Саму по себе зависимость найти не представляется возможным, но можно попытаться найти функцию (в разных задачах – алгоритм классификации или регрессионная модель), которая бы приблизила быy^* на всем множестве X (задача аппроксимации).

Каждый объект имеет признаки – отображения объекта на множества допустимых значений тех или иных признаков:

$$f_j: X \to D_j, \ j = 1 \dots p.$$

Если для объекта задан ряд признаков $f_1, \dots f_p$, то вектор $x = (f_1(x), \dots f_p(x))$ называется признаковым описанием объекта x, принадлежащего множеству объектов X.В решаемой задаче признаками являются различные факторы, которые могут влиять на успеваемость. Для каждого объекта-студента задан ряд факторов-признаков, являющийся признаковым описанием данного объекта.Признаки сами по себе разделяются на несколько типов. Каждый признак может быть:

- бинарным (либо он присутствует, либо отсутствует; например, студент может быть местным или иногородним);
- номинальным (принадлежит конечному множеству; например, форма обучения студента);
- порядковым (принадлежит упорядоченному конечному множеству; например, образование студента может быть среднее, бакалавр, магистр, и т.д.);

- количественным (принадлежит множеству действительных чисел; например, количество студентов, обучающихся в одной учебной группе с рассматриваемым студентом).

Данные о студентах, которые возможно получить, не являются формализованными, их необходимо привести к одному из типов признаков, чтобы в дальнейшем с ними можно было работать.После того, как определены объекты, и для каждого объекта определен ряд признаков, можно определить все имеющиеся данные в виде матрицы «объекты-признаки»

$$F = \begin{pmatrix} f_1(x_1) & \dots & f_p(x_1) \\ \dots & \dots & \dots \\ f_1(x_n) & \dots & f_p(x_n) \end{pmatrix}.$$

Строки в матрице соответствуют объектам (студентам), столбцы соответствуют признакам (формализованным значениям факторов). Матрица разнотипна, так как разные признаки могут иметь различные типы.

Что касается ответов, то, если их конечное множество, задача является задачей классификации, а именносопоставления объектам значений из конечного множества классов (возможно, классы могут пересекаться, одному объекту в данном случае, будут сопоставлены сразу несколько классов).Если ответ представляет собой непрерывную действительную величину, то задача является задачей регрессии.Если объекту соответствуют множество ответов с разной степенью релевантности, то задача называется задачей ранжирования.Для поставленной в работе задачи необходимо определить, какой тип задачи будет решаться, классификации или регрессии.

Моделью алгоритмов (регрессии или классификации) называется параметрическое семейство отображений A:

$$A = \{ a(x,w) \mid w \in W \},$$

где W – множество допустимых значений параметра w, и

$$a : X \times W \to Y$$

является фиксированной функцией (для фиксированного значения параметров).

Задачей является выбрать алгоритм из данного семейства, который бы лучше всего приблизил y^* на всем множестве X. Помочь в этом должна рассмотренная ранее обучающая выборка.

Методом обучения называется отображение вида

$$\mu : (X \times Y)^n \to A,$$

ставящее обучающей выборке X^n в соответствие некоторый алгоритм a из семейства A.

Можно сказать, что на данном этапе происходит обучение, то есть метод обучения строит по обучающей выборке алгоритм a:

$$a = \mu(X^n).$$

Удачно построенный алгоритм в дальнейшем для новых неизвестных ранее объектов (не входящих в обучающую выборку) должен будет суметь выполнить обратное действие – выдать ответ:

$$y = a(x).$$

Данный этап уже будет являться не обучением, а этапом применения.

Когда выбирается алгоритм из семейства по обучающей выборке, должна быть возможность оценивать выбранный алгоритм (можно сказать, оценивать параметры выбранного алгоритма).

Задача оценки алгоритма формализуется через введения функции потерь$L(a, x_i)$, которая выбранному алгоритму aи каждому объекту x_i из обучающей выборки ставит в соответствие целочисленное значение ошибки– измеренного некоторым способом расхождения между ожидаемым ответом y_i из выборки и полученным ответом $a(x_i)$ при использовании данного алгоритма. Таким образом, оценивается величина ошибки данного алгоритма на некотором объекте.

Примером оценки для задач классификации является функция потерь в виде индикатора ошибки (например, ошибка равна 0, если была выполнена правильная классификация и 1, если неправильная).

Для задачи регрессии примером является абсолютное значение ошибки:

$$L(a, x_i) = |y_i - a(x_i)|$$

или квадратичная ошибка:

$$L(a, x_i) = (y_i - a(x_i))^2.$$

Функция потерь высчитывается для каждого объекта из выборки, а для оценки работы алгоритма на всей выборки используется эмпирический риск:

$$Q(a, X^n) = \frac{1}{n}\left(\sum_{i=1}^{n} L(a, x_i)\right).$$

Чем ниже эмпирический риск для данного алгоритма и данной обучающей выборки, тем лучше данный алгоритм приближает искомую функцию y^* на множестве X.

Из этого можно сделать вывод, что для проведения лучшего обучения (выбора алгоритма по обучающей выборке) нужно решить задачу оптимизации: минимизации эмпирического риска, то есть подобрать такие параметры алгоритма, при которых эмпирический риск будет минимален:

$$W = argmin_{a \in A}\, Q(a, X^n),$$

где W – полученный в результате минимизации вектор параметров.

С другой стороны, одной из основных проблем машинного обучения является проблема переобучения, которая возникает при слишком малом эмпирическом риске на большой выборке: происходит ситуация, когда алгоритм идеально приближает искомую зависимость на обучающей выборке, но выдает большие потери на другой контрольной выборке, можно сказать, он становится «подогнан» конкретно под обучающую выборку.

Как итог, можно сказать, что для поставленной в работе проблемы необходимо выбрать тип решаемой задачи машинного обучения, осуществить выбор метода (классификации или регрессии) и построить модель на основе выбранного метода, которая позволила бы выбрать лучший алгоритм для решения задачи.

Для решения задач классификации и регрессии применяются разнообразные методы, такие как дерево принятия решения (логический метод классификации), некоторые метрические методы классификации, линейные методы классификации (метод стохастического градиента, логистическую регрессию и метод опорных векторов), линейная регрессия, нейронные сети, а также агрегированные алгоритмы классификации.

Для статистической обработки данных о студентах лучше всего использовать линейные регрессионные модели (гребневая регрессия, метод Лассо [6], эластичная сеть). Для рассматриваемых данных выбор такой моделей обусловлен, прежде всего, тем, что количество факторов или признаков может существенно превышать число наблюдений. Таким образом, количество параметров модели превышает число наблюдений, и использование нелинейной модели не имеет смысла, так как это приведет к еще большему увеличению параметров и большей чувствительности к шуму. Также преимуществом использования данных методов является то, что есть возможность получить вероятностную оценку для каждого ответа при классификации того или иного объекта. При этом для улучшения результатов желательно использование метода логистической регрессии с применением регуляризаторовL1 и L2. По результатам обучения алгоритма классификации будут получены и значения весов каждого признака, что даст ответ на вопрос, какие из признаков являются более весомыми, а какие играют меньшую роль при обучении.

Дляреализация модели обработки данных можно использовать язык программирования R [1], который направлен на статистическую обработку данных и был создан как альтернатива для языка S, имеющего аналогичные цели. Фактически, язык R в настоящее время стал стандартом для статистической обработки и различного анализа данных, часто используется для реализации алгоритмов машинного обучения.

Для реализации программной системы с помощью языка R можно использовать пакет glmnet, в котором реализованы алгоритмы построения моделей эластичной сети, гребневой регрессии и метода Лассо. Используя данный пакет, можно провести настройку регуляризаторов для данных методов, подобрав для них оптимальные значения.Исходными данными для программы должны являться текстовые файлы, с формализованными данными о студентах, в том числе, уровне их успеваемости, откуда будет производиться чтение. На основе этих данных производится обучение системы, то есть построение регрессионной модели и вычисление значений ее параметров. Результатами выполнения программы – выходными данными (текстовыми, как и входные), будут являться наборы наиболее важных с точки зрения влияния на успеваемость факторов.На основе полученных

знаний можно будет выполнять прогнозирование уровня успеваемости студентов.

Разработка системы прогнозирования на основе метода Лассо имеет широкий исследовательский потенциал. Полученные результаты и технологии могут быть применены в практике работы образовательных учреждений в качестве дополнительного источника прогнозной информации об абитуриентах и студентах при принятии решений о приеме и выборе специализации, а также непосредственно в учебном процессе. Управление в образовательной деятельности, как и управление в любой другой деятельности, предполагает прогноз и может быть осуществлено на его основе.

Библиографический список

1. The R Project for Statistical Computing: [сайт]. [2016]. URL: https://www.r-project.org/
2. Школа Анализа Данных (Яндекс): Машинное обучение: [сайт]. [2016]. URL: http://www.intuit.ru/studies/courses/13844/1241/info
3. Utkin L.V., Zhuk Y.A. A machine learning algorithm for classification under extremely scarce information // International Journal of Data Analysis Techniques and Strategies, Vol. 4, 2012. pp. 115-133.
4. Utkin L.V., Zhuk Y.A. Robust boosting classification models with local sets of probability distributions // Knowledge-Based Systems, Vol. 61, 2014. pp. 59-75.
5. Utkin L.V., Zhuk Y.A. Robust boosting classification models with local sets of probability distributions // Knowledge-Based Systems, Vol. 61, 2014. pp. 59-75.
6. Жук Ю.А., Егоров А.А., and Уткин Л.В. Математическая модель отбора параметров зимостойкости древесных растений на основе методов Лассо и эластичной сети // Нечеткие системы и мягкие вычисления (НСМВ-2014): труды Шестой всероссийской научно-практической конференции. СПб: Политехника-сервис, 2014. 213-222 pp.
7. Кошелева Г.В., Фионова Ю.Ю. Факторы, влияющие на успеваемость студентов // Актуальные направления научных исследований XXI века: теория и практика, Vol. 3, No. 7-4 (18-4), 2015. pp. 331-333.

М. О. Лебедев, кандидат технических наук, доцент

ОПТИМИЗАЦИЯ СТРУКТУРЫ ДИНАМИЧЕСКОЙ БД И РАБОТАЮЩЕГО С НЕЙ ПРИЛОЖЕНИЯ.

Под динамическими понимаются такие базы данных (БД), для которых состав фиксируемых параметров (состав полей и даже состав таблиц) не определен полностью на момент проектирования системы, или этот состав может меняться в ходе эксплуатации системы. Обобщенная структура

такой БД на логическом уровне в большинстве случаев может быть представлена схемой, показанной на рис. 1.

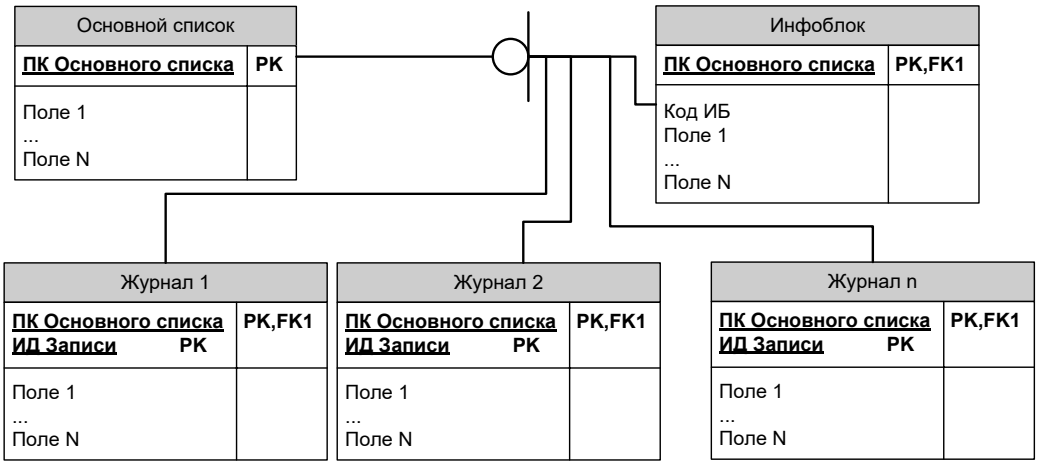


Рис. 1. Обобщенная схема динамической БД

Таблица "Основной список" содержит список основных объектах, для работы с которыми используется данная система. Состав полей этой таблицы (параметров или атрибутов объектов) определяется предметной областью, для которой предназначена данная информационная система. Поле "ПК Основного списка" - первичный ключ таблицы. В качестве первичного ключа целесообразно использовать одно поле, каждое значение которого уникально. В зависимости от использования информационной системы это может быть поле типа "счетчик" (целочисленное поле, значение которого автоматически увеличивается при добавлении новой записи), или иметь более сложный алгоритм определения нового значения (например, значения на базе GUID - глобального уникального идентификатора). Тип "счетчик" обеспечивает уникальность первичного ключа как для desktop'ных (настольных, или однопользовательских) БД, так и для клиент-серверных корпоративных БД, если значение первичного ключа определяется на сервере. Если информационная система представляет собой совокупность независимых баз данных (и приложений, работающих с этими базами), данные которых в последствии должны быть объединены в некую общую базу, то в качестве первичного ключа целесообразно использовать значения на базе GUID. Основные требование к составу полей "Основного списка":

- обязательное наличие этих атрибутов у всех объектов, информация о которых хранится в БД;
- эти атрибуты являются фактическими данными (т.е. не требуют сохранения истории изменения).

Набор данных "Инфоблок" содержит атрибуты (поля), также характеризующий каждый объект списка. Но, во-первых, состав этой информации определяется назначением информационной системы, а, во-вторых, эти атрибуты не обязательно должны быть у каждого объекта. Атрибуты

"Инфоблока" также являются фактическими данными. На логическом уровне "Основной список" и "Инфоблок" можно представить как две таблицы, имеющие связь "один-к-одному". Не редко состав полей информационного блока (набора данных "Инфоблок") настолько объемный, что пользователю бывает затруднительно воспринимать представленную в нем информацию. В этом случае целесообразно группировать такую информацию по каким-либо признакам. Например, блок информации "О родителях" (если ИПРС связана с регистрацией учащихся), или "Перенесенные заболевания" (если информационная система о пациентах). Именно с этой целью в набор данных "Инфоблок" внесено поле "Код ИБ" (код информационного блока). Для каждого блока информации м.б. свой состав полей. Но все поля Инфоблока логически представляют собой одну таблицу.

Совокупность журналов (с 1-го по *n*-й) на логическом уровне представляют собой таблицы, каждая из которых имеет свой состав полей (как по количеству, так и по типам данных). Но все эти таблицы логически имеют отношения с таблицей "Основной список" связь "один-ко-многим". Т.е. одной записи в таблице "Основной список" может соответствовать больше одной записи в каждом журнале.

Переходя от абстракции "объект" к абстракции "данные", т.е., выполняя нормализацию таблиц БД не на уровне характеристик объектов, а на уровне типов данных, описывающих эти характеристики, структуру таблиц и информационного блока, и журналов может быть представлена следующим образом (рис. 2).

Рис. 2. Структура информационных таблиц динамической БД

Здесь, "Журнал *I*" - таблица, хранящая данные *i*-го типа (*i* - строковый тип, целочисленный тип, вещественный тип, логический тип и т.д.); "Код_поля" - определяет логическое имя поля, его тип данных, и способ ввода значения (вручную или выбирая значение из списка - из справочника); "Код_журнала" - идентификатор журнала на логическом уровне (см. рис. 1), в котором должны находиться эти данные (на логическом уровне); "Код_записи" - поле, однозначно определяющее запись в логических журналах или очередное значение параметра в информационном блоке (если параметр на логическом уровне допускает множественность значений); "ПК Основного списка" - поле (или состав полей), определяющий к какому объекту основного списка относится это значение. В таблицах такой структуры хранятся данные, которые на логическом уровне могут быть представленные как в журналах, так и в информационном блоке. Локали-

зация данных (принадлежность к конкретному журналу/информационному блоку) определяется полем "Код_журнала". В этом случае каждый информационный блок, который определялся значением поля "Код ИБ", в данном случае представляет отдельный журнал, т.е. определяется значением поля "Код_журнала". Для такой структурой данных, необходимо иметь описание принадлежности полей к тому, или иному журналу. Это вопрос решается с помощью создания таблиц, описывающих варианты журналов (динамический журнал, или квазистатический журнал - информационный блок), и составы полей, относящиеся к логическим журналам. На рис. 3 показана структура таблиц, описывающих эти метаданные.

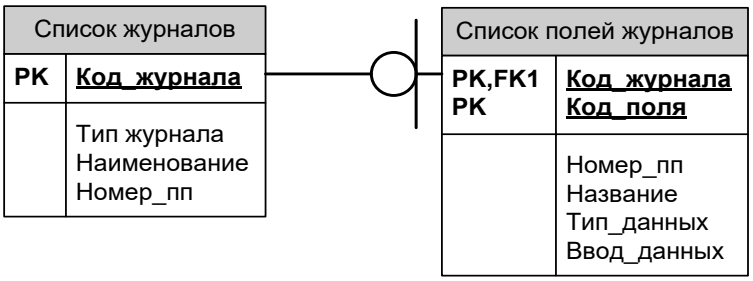


Рис. 3. Структура таблиц, определяющая состав журналов динамической ИПРС

Подобная схема организации данных не требует перепрограммирования ни сервера БД, ни клиентского приложения (для отображения, редактирования и поиска информации) при изменении состава параметров и журналов информационной системы. Однако непосредственное использование такой структуры данных приводит к значительному увеличению объема БД из-за наличия ключевых полей, объединяющих данные в отдельную запись журнала. Рассмотрим следующий пример. Пусть какой-либо логический журнал информационной системы содержит следующие информационные поля:

Дата	Min значение	Max значение	Примечание
Тип: Дата/ ДатаВремя	Тип: Integer	Тип: Integer	Тип: строка
Размер 8 байт	Размер 4 байта	Размер 4 байта	Размер 255 байт

На логическом уровне связь основного источника и журнала *I* для классической структуры данных (абстрагирование на уровне объекта) будет иметь вид, показанный на рис. 4.

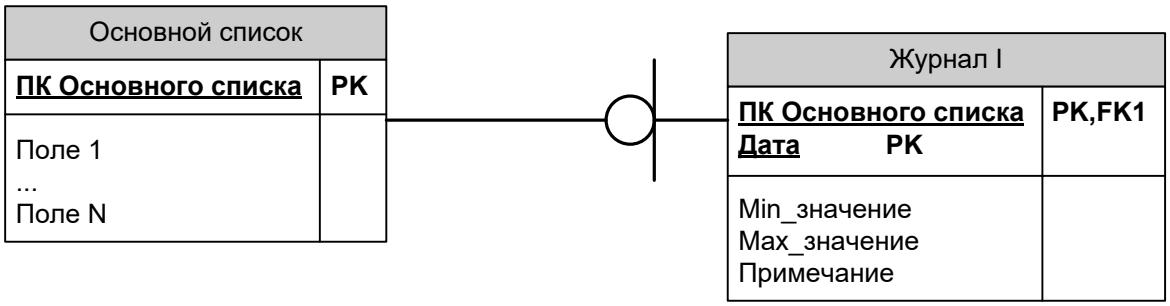


Рис. 4. Классическая структура данных основного списка и журнала

Пусть "ПК Основного списка" представляет собой поле типа "счетчик", размер которого 4 байта. Тогда на одну запись в журнал потребуется 275 байт. При использовании динамической структуры данных (абстрагировании на уровне типа данных) одна запись такого журнала будет храниться в трех различных таблицах:

1. таблице, хранящей данные типа "Дата" (поле "Дата");
2. таблице, хранящей данные типа "Целое число" (поля "Min значение" и "Max значение");
3. таблице, хранящей данные типа "Строка" (поле "Примечание").

Причем, значение каждого поля предваряется значениям 4-х ключевых полей: "ПК Основного списка", "Код_журнала", "Код_поля" и "Код_записи". Пусть последние 3 поля также имеют тип данных "Целое число" (размер 4 байта; будем полагать, что это не распределенная, а корпоративная информационная система, в которой значения всех ключевых полей определяются на сервере). Таким образом, для хранения одной записи журнала в данном случае потребуется 4×4+8+(4×4+4)×2+ 4×4+255 = 335 байт. Что примерно на 22% увеличивает объем, необходимый для хранения одной записи в журнале, по сравнению с классическим подходом. Еще большее увеличение объема данных будет в случае распределенной информационной системы, когда значения первичных ключей будут формироваться на базе GUID. В общем случае, если журнал содержит N полей, то дополнительный объем данных в динамической системе на одну запись будет составлять 3×S×N байт (здесь S - размер полей "Код_журнала", "Код_поля" и "Код_записи"). Соответственно, для R записей дополнительный объем будет равен 3×S×N×R байт. Так, для рассмотренного примера S=4, N=4. Тогда, для 100 записей журнала дополнительный объем составит 3×4×4×100=4800 байт.

Альтернативой такой организации данных в сочетании с обеспечением динамичности БД может быть создание журналов классической структуры на основе метаданных, описывающих эти журналы. При этом имена полей этих квазиклассических журналов могут формироваться по определенным правилам, включая в себя значения кода поля, указанного в метаданных при описании журналов. Для статических журналов (инфоблоков) сохраняется абстрагирование на уровне типов данных. В этом случае увеличение объема БД будет в пределах 18 - 37% по сравнению с классической структурой данных. Тем самым достигается компромисс

между обеспечением гибкости (динамичности) структуры БД и ее размером.

Для приложения, работающего с такой БД, должны быть сформулированы определенные правила, касающиеся отображения данных. Эти правила определяют алгоритм работы с данными (отображение данных, редактирование данных, поиск данных). Формулировки этих правил зависят от предметной области, для которой создается информационная система, и возможностями среды разработки приложения (возможностями визуальных/невизу-альных компонентов, обработчиков событий и т.п.). Очевидно, что эти правила должны обеспечивать добавление различных журналов и полей инфоблоков, не требуя перепрограммирования приложения, за исключением, возможно, блока, отвечающего за статистическую обработку. Такой блок может быть выполнен в виде dll-компонента. Тем самым будет обеспечена бесперебойная работа приложения при изменении логической структуры данных. Изменение логической структуры данных позволяет без перепрограммирования приложения выполнять работу с данными (добавлять новые записи, редактировать существующие, удалять, осуществлять поиск/выборку данных). А по мере необходимости dll-компонент, отвечающий за статистическую обработку может быть перепрограммирован под учет новых логических полей и передан в соответствующие службы (на определенные рабочие места).

Н.В. Лушкин, кандидат технических наук, доцент
Н.П. Васильев, кандидат технических наук, доцент

МОДЕЛИРОВАНИЕ ДРЕВЕСНЫХ СТРУКТУР МНОГОУГОЛЬНИКАМИ ВОРОНОГО (ЯЧЕЙКАМИ ДИРИХЛЕ).

Введение

В статье обсуждается задача моделирования графической структуры многоугольниками Вороного (ячейками Дирихле), что расширяет представление о древесине как природном конструкционном материале. Многоугольники Вороного получаются при разбиении плоскости, на которой заданы произвольные точки, на соответствующее количество областей. Каждая такая область - это множество точек плоскости наиболее близких к соответствующей точке *рис.1*. Пусть задана плоская графическая структура (например, *рис. 2*). Построить наиболее близкую модель Вороного, для графической древесной структуры *рис.3*.

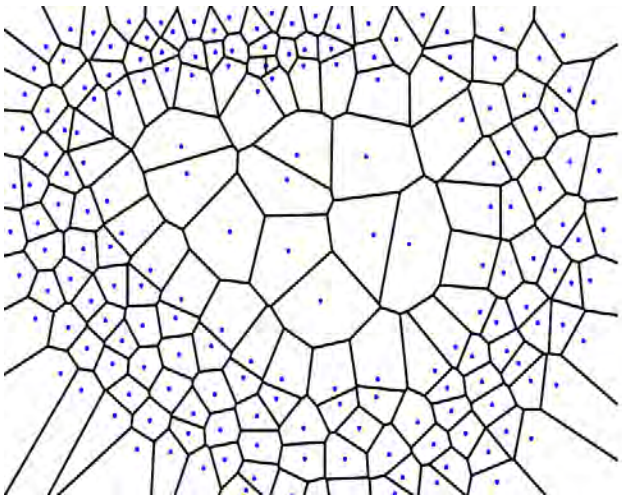


Рис.1. Ячейки Вороного для множества точек на плоскости

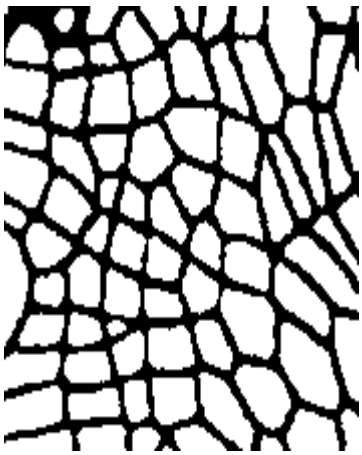


Рис.2. Пример древесной структуры.

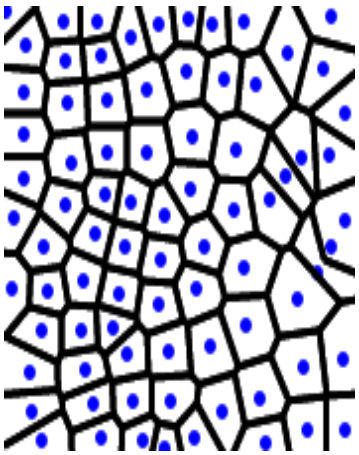


Рис.3. Многоугольники Вороного

Для решения поставленной задачи моделирования древесных структур многоугольниками Вороного, кратко опишем последовательность действий. На графическом изображении определяется множество связанных объектов. Определяются координаты центров тяжести каждого объекта. Принимая эти координаты, как по множества точек на плоскости, используя алгоритм Форчуна. Реализованному в [2], строятся многоугольники Вороного.

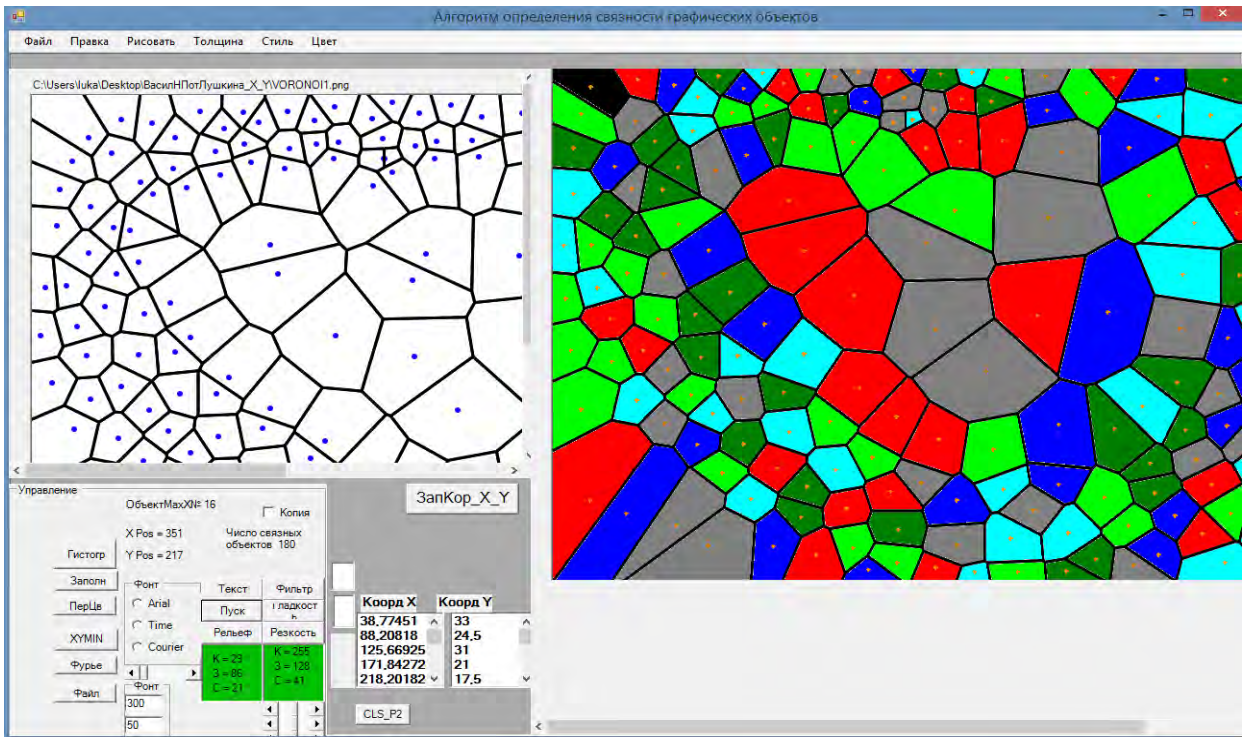


Рис. 4. Расчет координат (x , y) центров тяжести связных объектов

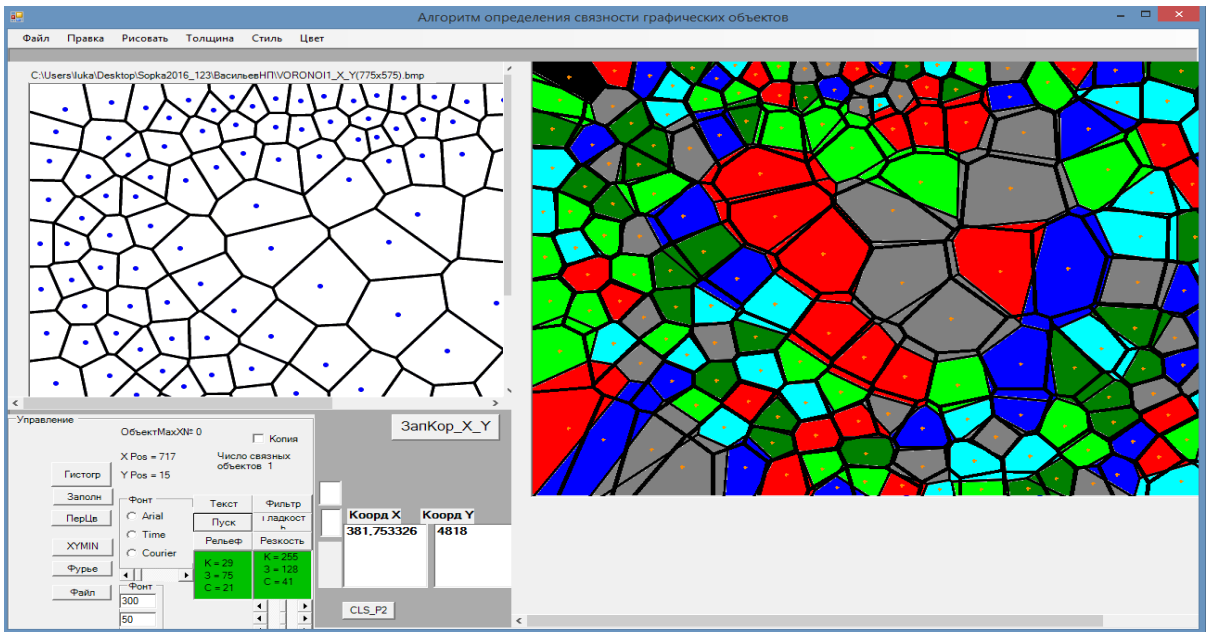


Рис.5. Сопоставление графического объекта, полученного по координатам (x ,y) центров тяжести, с исходной моделью Вороного

Если многоугольники Вороного (ячейки Дирихле) (рис. 3) рассматривать как планарный граф, то уравнение Эйлера для графов иметь вид:

$$b - n = p_1 - p_0,$$

где p_0 - нулевое число Бетти(число компонент связности графа), p_1 - первое число Бетти(число граней графа), b - число вершин, n - число рёбер.

При построении многоугольников Вороного, число вершин (p) и число ребер (n) графа, определяются по модели графического объекта.

Результаты моделирования древесных структур (рис. 6 – 18).

Структура льна

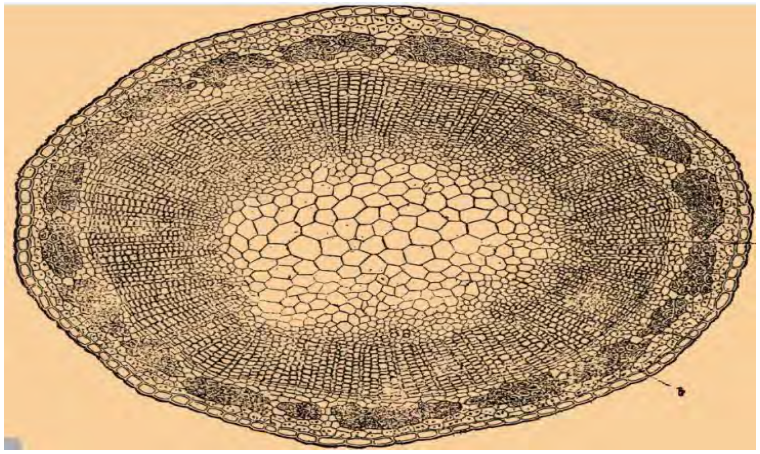


Рис.6. Структура льна

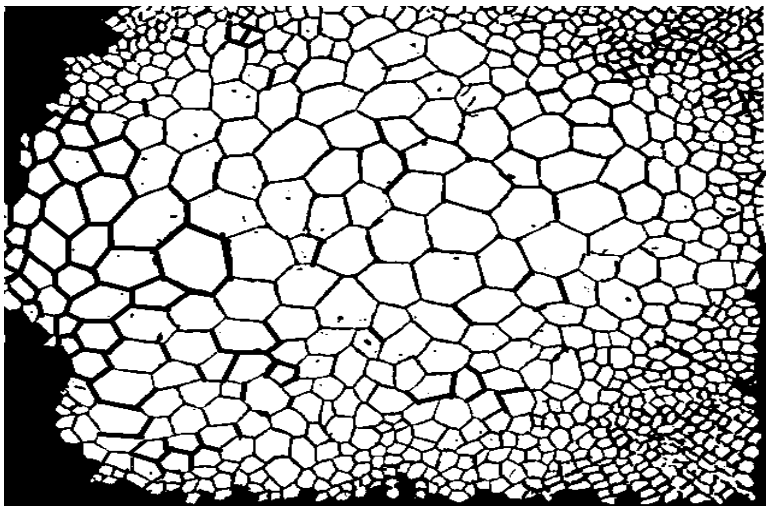


Рис 7. Выделенная область структуры льна

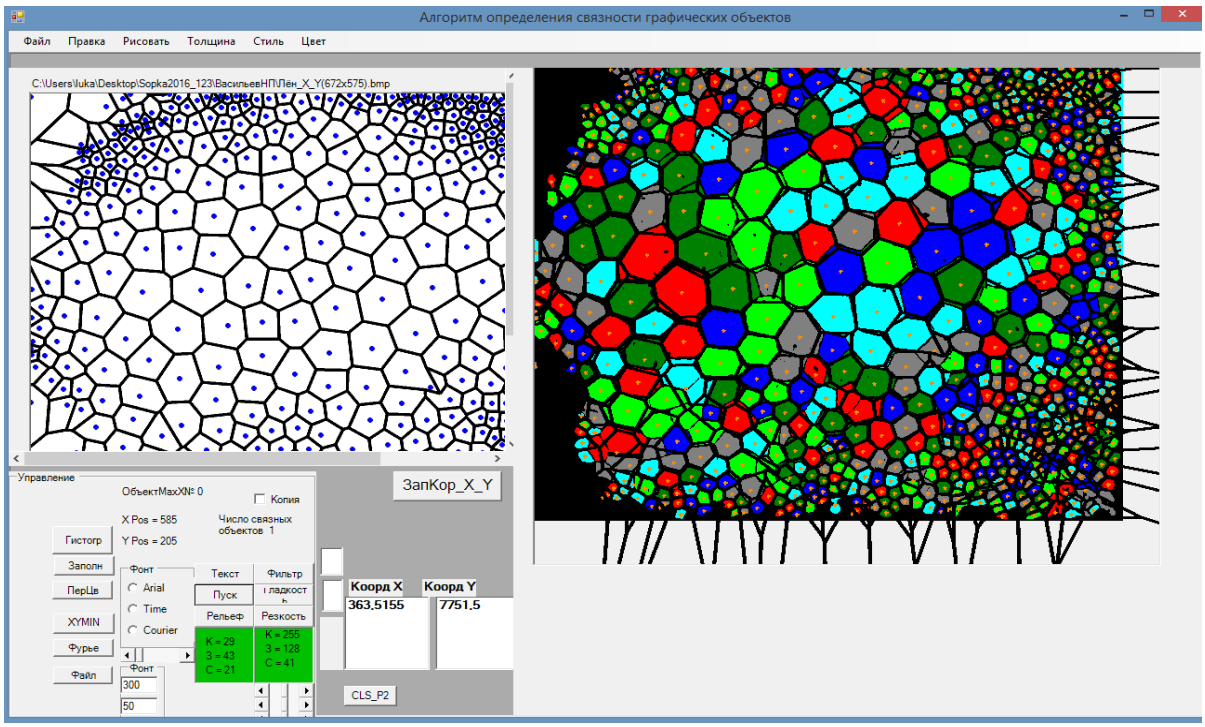


Рис.8. Сопоставление графического объекта (многоугольников Вороного), полученного по координатам (x ,y) центров тяжести, с исходной структурой льна.

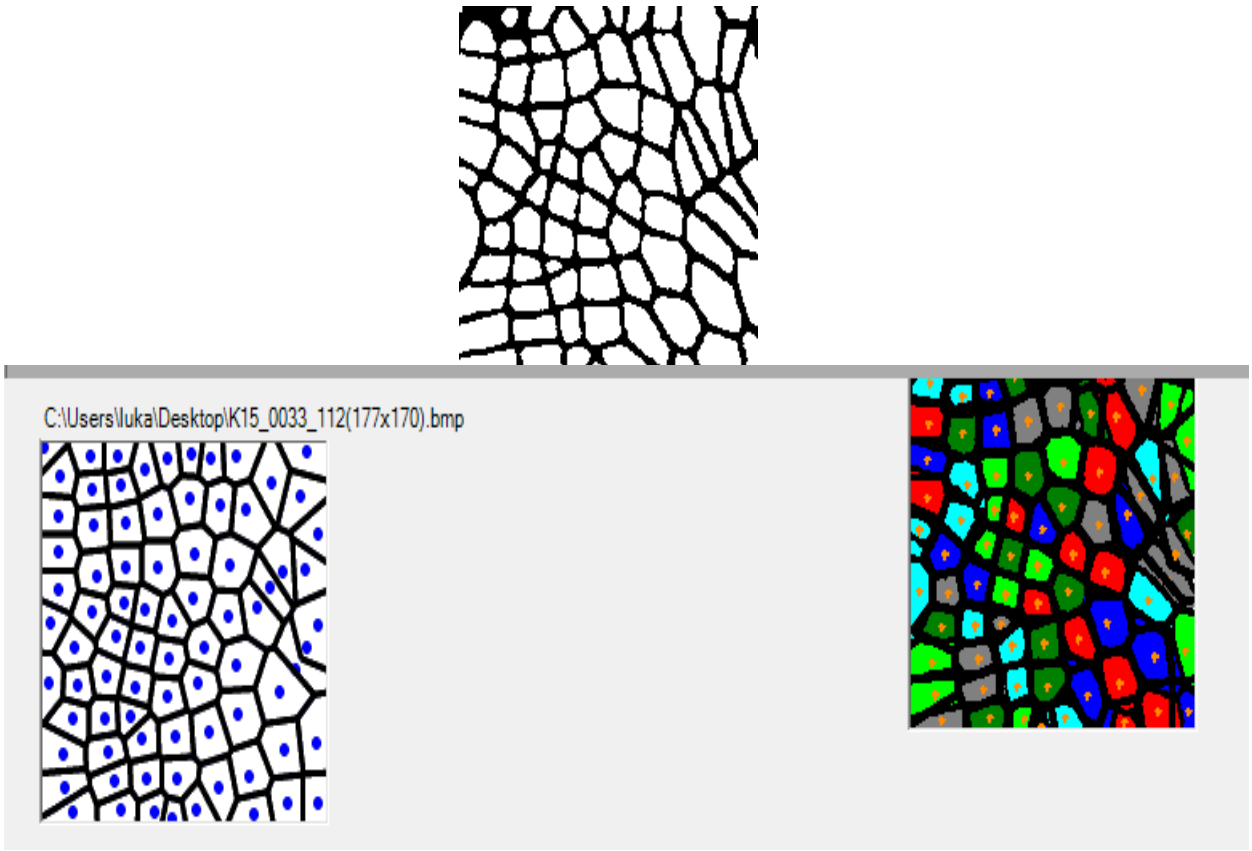


Рис.9. Пример моделирования древесной структуры многоугольниками Вороного

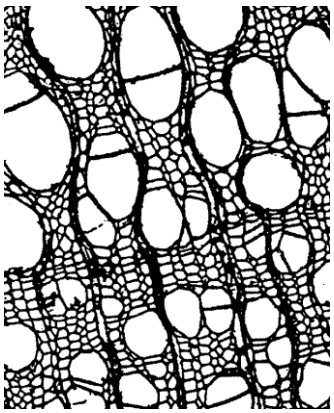


Рис. 10. Угловатые, разновеликие, тонкостенные волокнистые элементы в древесине ивы

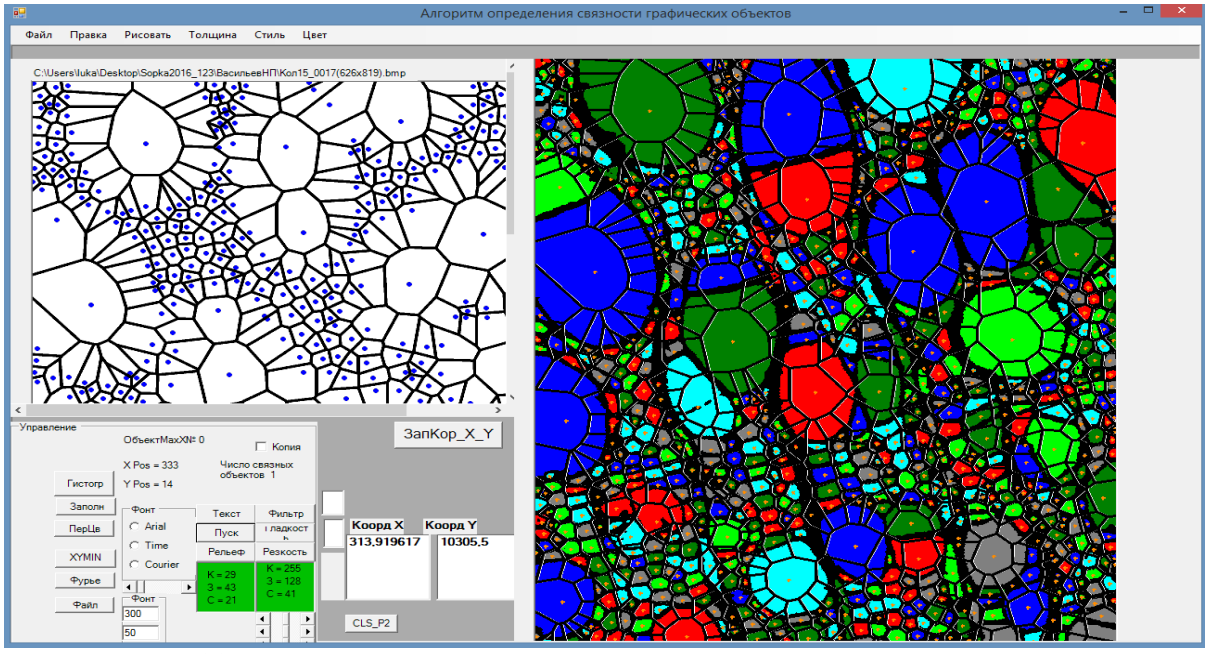


Рис.11. Моделирование структуры ивы многоугольниками Вороного

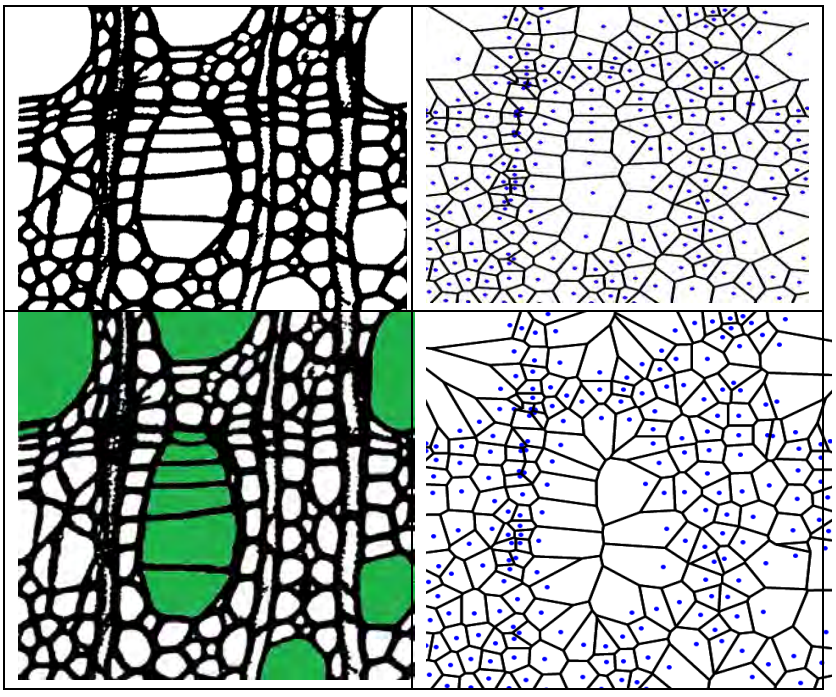


Рис.12. Примеры моделирования древесных структур

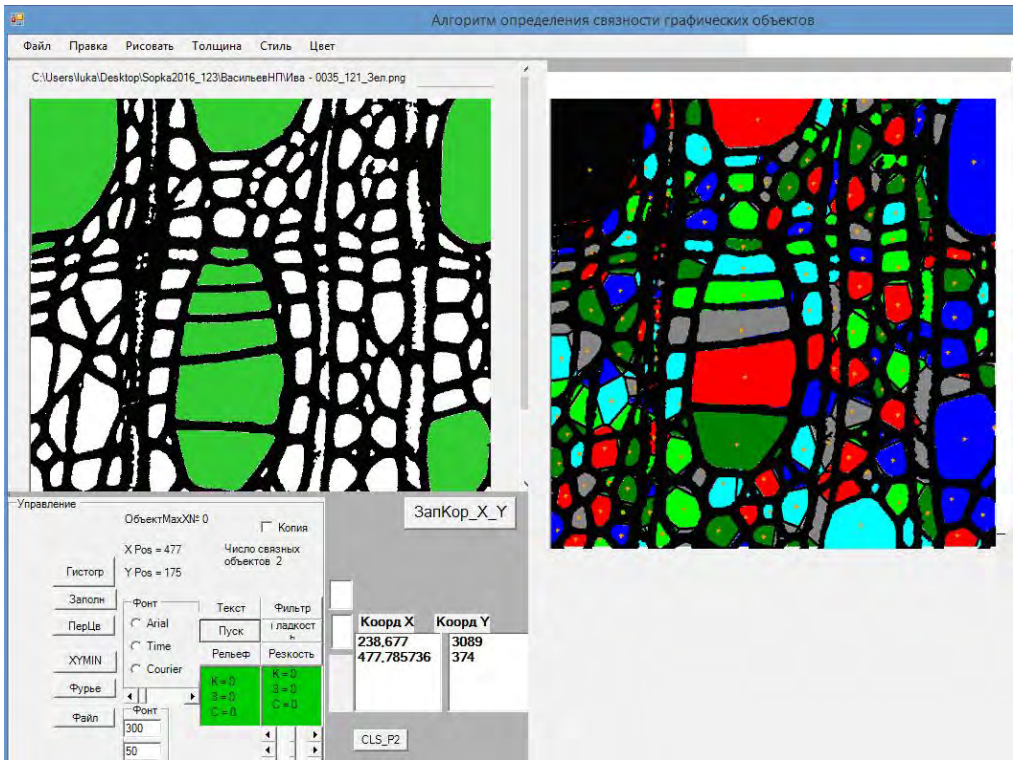


Рис. 13. Сопоставление графического объекта (многоугольников Вороного), полученного по координатам (x ,y) центров тяжести, с исходной структурой ивы

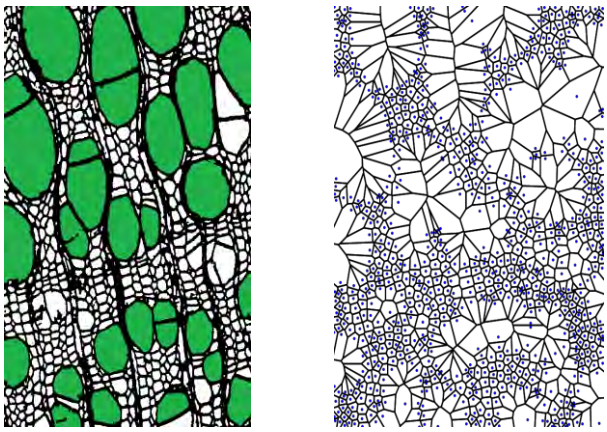


Рис. 14. Пример моделирования древесных структур

Структура ели

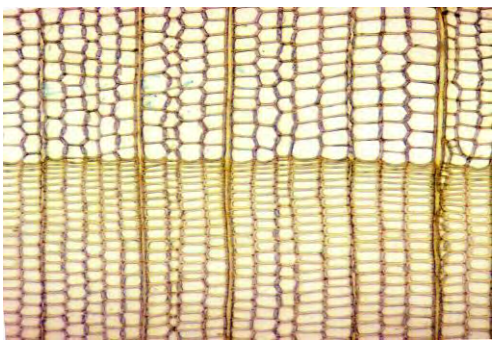


Рис.15. Структура ели

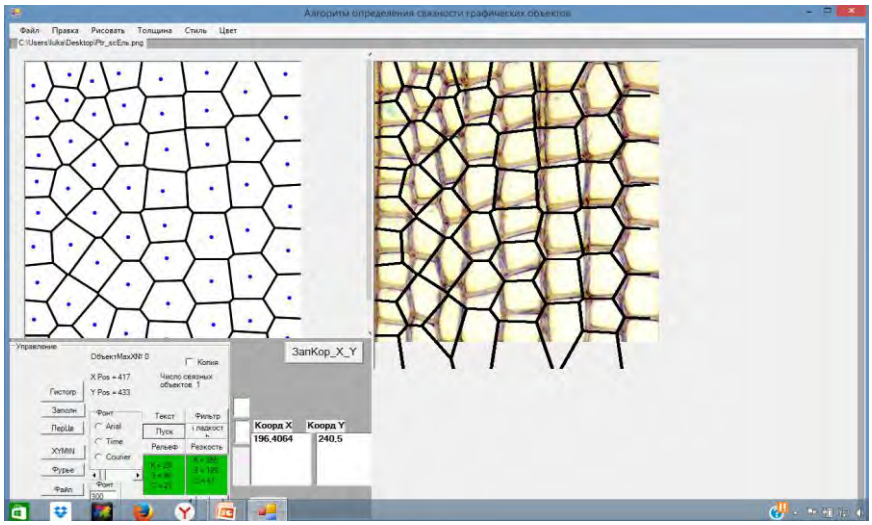


Рис.16. Сопоставление графического объекта (многоугольников Вороного), полученного по координатам (x ,y) центров тяжести, с исходной структурой ели

Моделирование микробиологических структур(колоний бактерий) многоугольниками Вороного

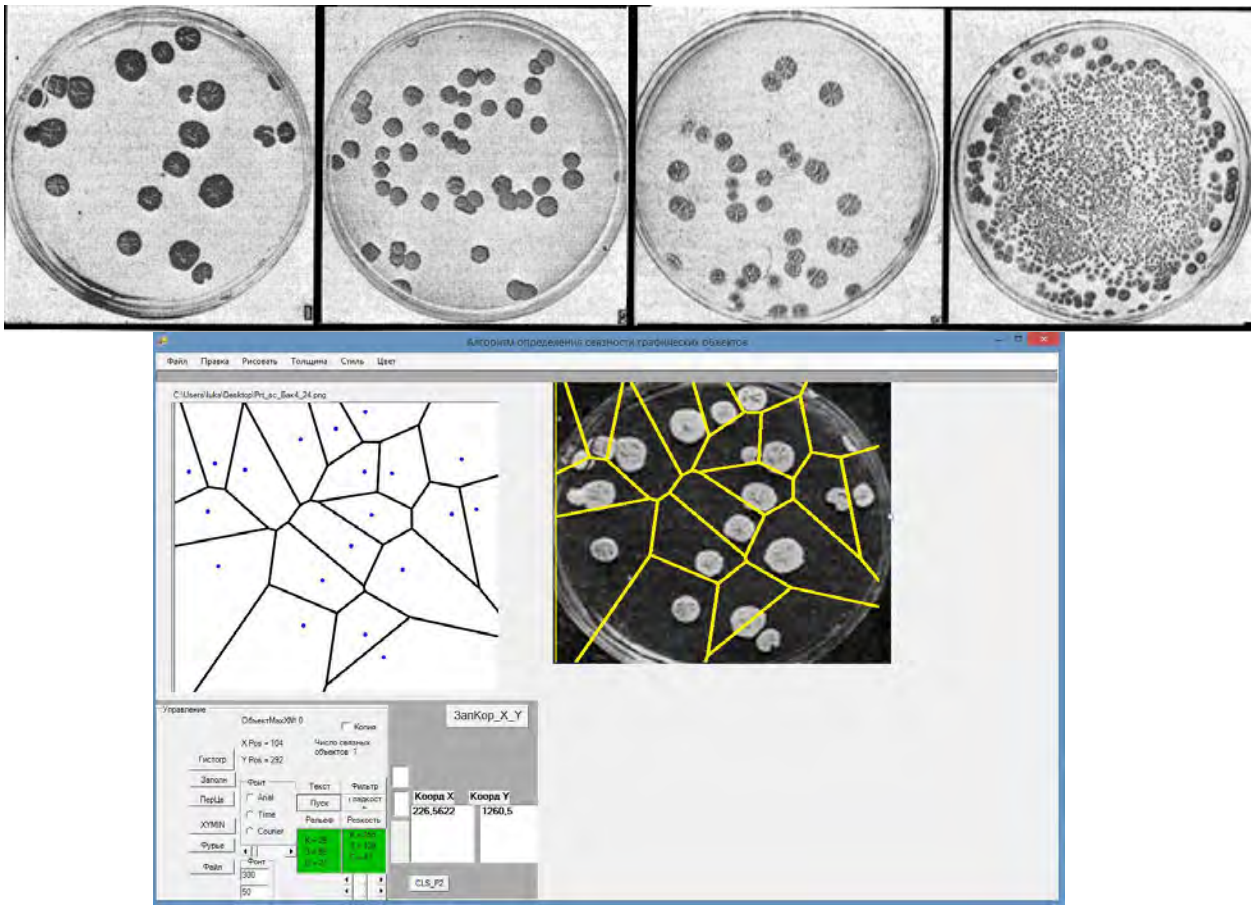


Рис.17. Моделирование микробиологических структур(колоний бактерий) многоугольниками Вороного

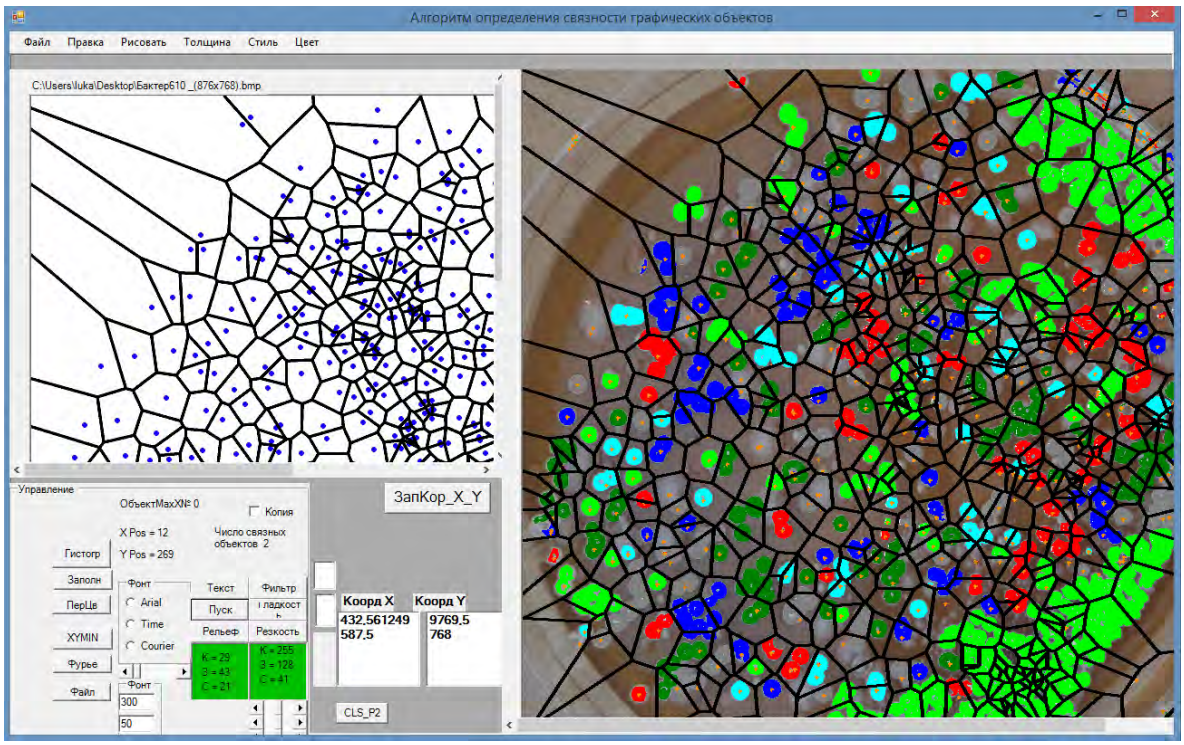


Рис. 18. Моделирование микробиологических структур(колоний бактерий) многоугольниками Вороного

Библиографический список

1. Колосова М. И., Соловьева Н. Г. Основные анатомические признаки древесины лиственных деревьев и кустарников. СПбГЛТУ, 2013.

2. Васильев Н.П. Реализация алгоритма Форчуна расчета диаграммы Вороного на PHP. В сб. "Информационные системы и технологии: теория и практика" Сб. науч. Трудов. Вып.7 СПб, СПбГЛУ, 2015. с. 10-16

3. Лушкин Н.В. Алгоритм определения связности графических объектов. - Труды Санкт-Петербургской Государственной лесотехнической академии Актуальные проблемы развития высшей школы Санкт-Петербург. 2010. Стр. 308-309.

Н.В. Лушкин, кандидат технических наук, доцент
В.Б.Бочкова, магистр

ОПРЕДЕЛЕНИЕ НЕКОТОРЫХ ПАРАМЕТРОВ МИКРОБИОЛОГИЧЕСКИХ ОБЪЕКТОВ ПО ИХ ГРАФИЧЕСКИМ ФАЙЛАМ

Микробиология — наука, изучающая строение, жизнедеятельность и экологию микроорганизмов — мельчайших форм жизни растительного или животного происхождения, не видимых невооруженным глазом. Мик-

робиология изучает всех представителей микромира (бактерии, грибы, простейшие, вирусы).

В микробиологической практике часто необходимо не только выделить тот или иной вид, но и определить численность живых микроорганизмов в тех или иных объектах исследования. Для этого используются различные методы количественного учета.

Описание методов количественного учета.

Наиболее распространенным является метод серийных разведений. Этот метод позволяет вести учет только жизнеспособных клеток микроорганизмов. Вторым методом является – метод подсчета микроорганизмов под микроскопом в счетной камере Горяева. Этот метод подсчета является менее достоверным т.к. при подсчете считаются как жизнеспособные так и не жизнеспособные клетки.

Метод предельного разбавления образца (метод серийных разведений) позволяет с высокой точностью определить количественное значение группы микроорганизмов.

Сущность методики заключается в том, что исследуемая проба неоднократно разводится стерильной водой определенным образом. Из полученных разведений делают пересев на плотную питательную среду с агаром в чашки Петри.

Засеянные чашки Петри термостатируют при температуре, оптимальной для исследуемых микроорганизмов и через определенное время проводят подсчет выросших колоний.

В данной работе микробиологическим объектом исследования является бактерии рода (Actinomyces).

Актиномицеты — микроорганизмы, имеющие признаки и бактерий, и грибов. По строению и биохимическим свойствам актиномицеты аналогичны бактериям, а по характеру размножения, способности образовывать гифы и мицелий похожи на грибы.

Актиномицеты — микроорганизмы, занимающие промежуточное место между грибами и бактериями. По строению и биохимическим свойствам актиномицеты аналогичны бактериям, а по характеру размножения, способности образовывать гифы и мицелий похожи на грибы.

Чрезвычайно широкое распространение актиномицетов в природе дает основание полагать, что этим организмам принадлежит большая роль в круговороте веществ, как органических, так и минеральных.

Актиномицеты при помощи биотехнологий при культивировании на питательных средах в процессе биосинтеза выделяют химические вещества, которые имеют высокую инсектицидную (против насекомых), фунгицидную (против грибов) и антибактериальную активность.

Из многих культур актиномицетов получают лекарственные препараты - антибиотики, используемые в медицине, ветеринарии, растениеводстве, пищевой промышленности. Актиномицеты способны продуцировать витамины, гормоны, ферменты, токсины, ростовые вещества, аминокислоты и другие полезные для человека биологические вещества. Важ-

ную роль актиномицеты играют в процессах почвообразования и формировании плодородии почв.

Общие замечания по методам серийных разведений.

Несмотря на то, что методы серийных разведений являются наиболее точными и информативными, их постановка в практических лабораториях сопряжена со значительными трудностями. Прежде всего, речь идет о необходимости ручного подсчета КОЕ (колониеобразующая единица) каждой чашки Петри с исследуемым образцом, что сильно сказывается на времени проведении исследования. Помимо этого при изобилии загрязнения или КОЕ (колониеобразующая единица), ручной подсчет практически невозможен, что в дальнейшем приводит к повторному проведению опыта. Также необходимо учитывать погрешность, связанную с человеческим фактором.

Иллюстрация примеров бактерий (Актиномицетов) при серийном разведении.

На рисунке 1 мы наблюдаем изобилие КОЕ, где ручной подсчет колоний практически невозможен.

На рисунке 2 мы наблюдаем изобилие контаминации (загрязнения).

На рисунке 3 мы видим стандартный вид КОЕ.

Рис.1. Изобилие КОЕ. **Рис.2.** Изобилие контаминации. **Рис. 3**. Стандартный вид.

Необходимость перехода к компьютерным вычислениям.

Несмотря на то, что методы серийных разведений являются наиболее точными и информативными, их постановка в практических лабораториях сопряжена со значительными трудностями.

Прежде всего, речь идет о необходимости ручного подсчета КОЕ (колониеобразующая единица) каждой чашки Петри с исследуемым образцом, что сильно сказывается на времени проведении исследования.

Помимо этого при изобилии контаминаций или КОЕ (колониеобразующая единица) , ручной подсчет практически невозможен, что в дальнейшем приводит к повторному проведению опыта. Также необходимо учитывать погрешность, связанную с человеческим фактором.

Испытав теорию подсчета КОЕ, при помощи программы, нами были получены результаты сопоставимые с ручным подсчетом.

Описание алгоритма определения количества бактерий.

Изображения бактерий могут быть как изолированными, так, и объединены в группы связных областей по несколько бактерий (*рис.1.,2.,3.*). Ставится задача оценить количество бактерий находящихся на изображении графического файла. Предлагается следующие этапы решения поставленной задачи:

1. отмечаем в левом окне *(рис.4*) цвет в исходном изображении (цвет бактерий) ;
2. включаем кнопку "ПУСК";
3. программа на правом окне изображает связные объекты(колонии бактерий) различными цветами;
4. последовательно для каждого связанного объекта программа определяет площадь и отображает на информационном окне;
5. последовательно отмечаем небольшие объекты(колонии бактерий), где возможно малое количество бактерий (до 10);
6. программа восстанавливает исходный цвет отмечаемого объекта и просит ввести количество наблюдаемых бактерий (*рис.5.*);
7. программа определяет среднюю площадь бактерии;
8. в зависимости от исходного графического файла и опыта пользователя, пункты 5, 6, 7 выполняем несколько раз;
9. отказываемся от ввода количества бактерий;
10. программа показывает количество отмеченных бактерий и рассчитывает среднюю площадь бактерии для вычисления общего количества бактерий;
11. включаем кнопку "РасчБак", чтобы на информационном окне отобразить общее количество бактерий в исходном графическом файле.

Выводы

Данная компьютерная обработка колоний Actinomyces хорошо согласуется с результатами сопоставимые с ручным подсчетом.

Заявленный подход при обработке графических файлов микробиологических объектов, позволяет выявить актуальные направления и обратить внимание на развитие компьютерного анализа характеристик микробиологических объектов.Программа позволяет создать систему оперативной обработки и оценки характеристик колоний Actinomyces.

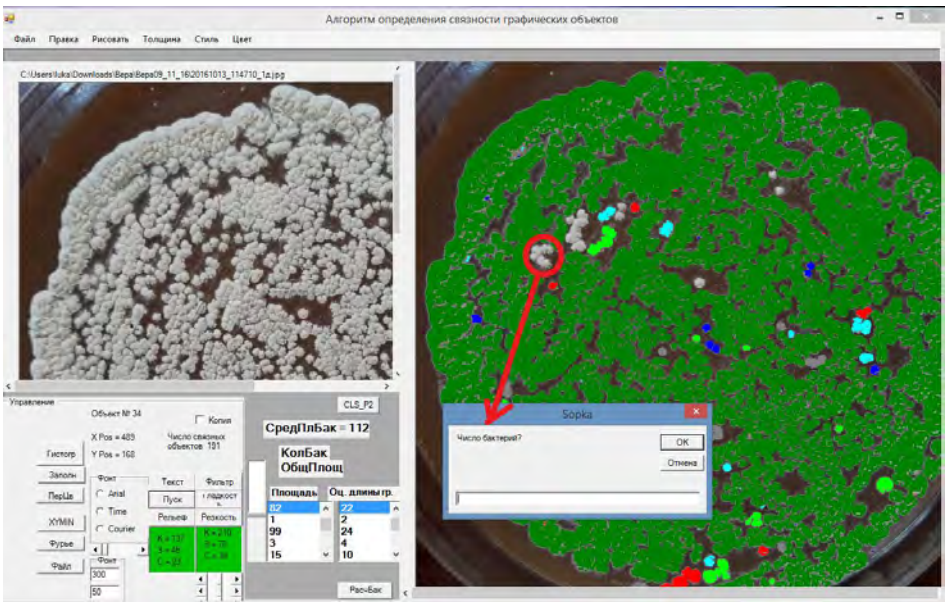


Рис.4. Интерфейс программы (Демонстрация ручного ввода количества бактерий)

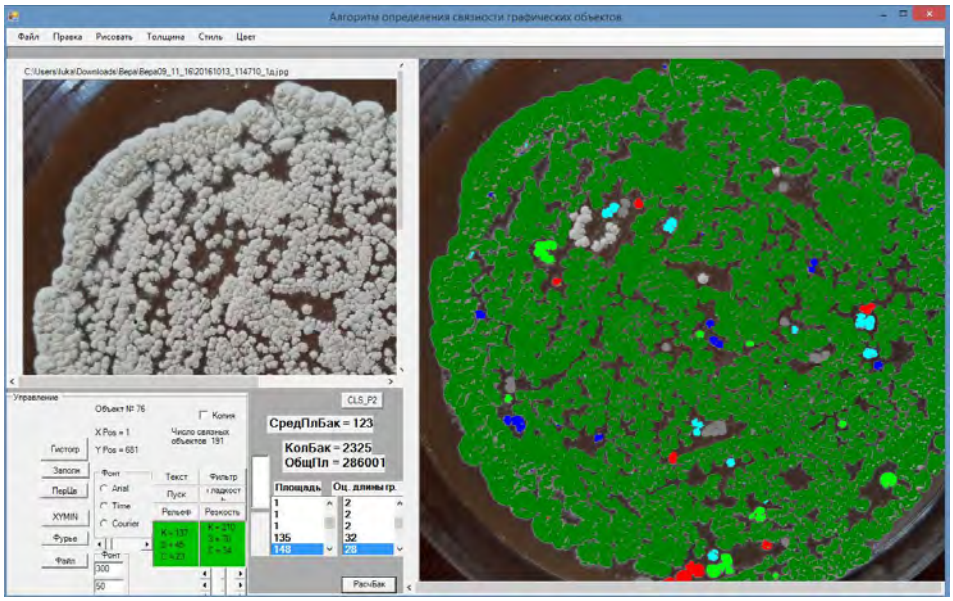


Рис.5. Демонстрация расчета общего количества бактерий при больших колониях бактерий

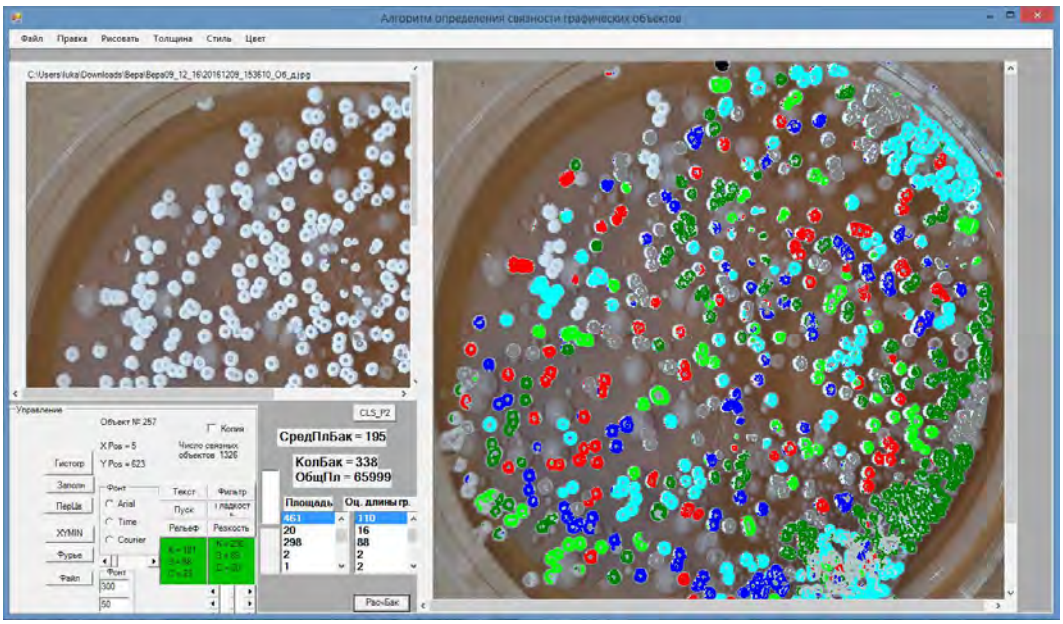


Рис.6. Демонстрация расчета общего количества бактерий при небольших колониях бактерий

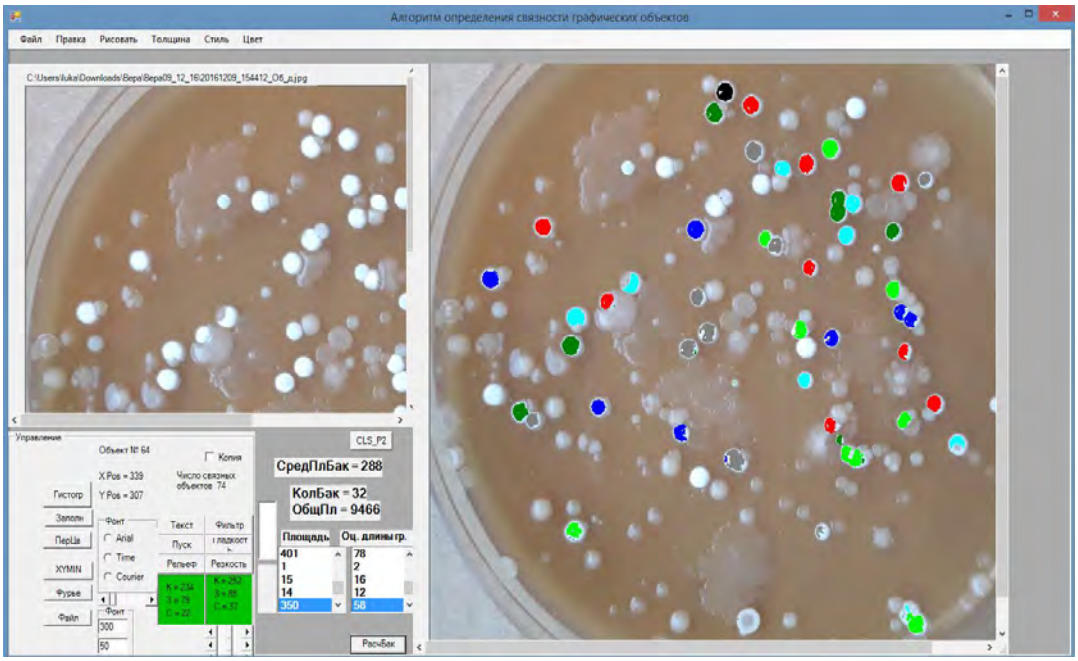


Рис.7. Демонстрация расчета общего количества бактерий при малых колониях бактерий

Библиографический список

1. Жизнь растений. Том 1. Введение. Бактерии и актиномицеты' \\Под редакцией члена-корреспондента АН СССР профессора Н. А. Красильникова и профессора А. А. Уранова - Москва: Просвещение, 1974 - с.487

2. Дикий И.Л., Сидорчук И.И., Холупяк И.Ю. и др. Микробилогия: Руководство к лабораторным занятиям: Учебное пособие для студентов фар-

мацевтических ВУЗов и фармацевтических факультетов медицинских институтов. // К.: ИД “Профессионал”, 2004. – С

3. Руководство к практическим занятиям по медицинской микробиологии и лабораторной диагностике инфекционных болезней /Под ред. Проф. Кривошеина Ю.С. – К.: Вища школа, 1986. – С. 25-31

4. Лушкин Н.В. Алгоритм определения связности графических объектов. - Труды Санкт-Петербургской Государственной лесотехнической академии Актуальные проблемы развития высшей школы Санкт-Петербург. 2010. Стр. 308-309.

А.А. Никифоров кандидат с-х. наук, доцент

ОБРАБОТКА МАТЕРИАЛОВ АЭРОФОТОСЪЕМКИ, ПОЛУЧЕННЫХ С ПОМОЩЬЮ БЕСПИЛОТНОГО ЛЕТАТЕЛЬНОГО АППАРТА

Для изучения лесного растительного покрова применяются различные методы, наиболее широкое применение получили дистанционные методы. Они основаны на получении информации об исследуемых объектах на расстоянии путем регистрации электромагнитных излучений при помощи чувствительных приемников, установленных на космических кораблях, самолетах, вертолетах, беспилотных летательных аппаратах [Сухих, 2005, Заяц 6,7]. Беспилотный летательный аппарат (БПЛА) представляет собой разновидность радиоуправляемого летательного аппарата, управление которым осуществляется оператором с земли, или в автоматическом режиме (на автопилоте) [Зинченко, 2011].

Наиболее распространенными беспилотными летательными аппаратами являются БПЛА самолетного и вертолетного типа. У каждого типа есть свои преимущества и недостатки. БПЛА самолетного типа способны выполнять полет длительное время. Для взлета и посадки необходимо открытое пространство, либо требуются дополнительные механизмы – катапульта, парашют. БПЛА вертолетного типа, наоборот, для взлета и посадки не требуется много места, но продолжительность полета в разы меньше. Согласно классификации, многороторные платформы относятся к беспилотным летательным аппаратам вертолетного типа [Зинченко, 2011].

Исследования выполнены на примере Лисинского учебно-опытного лесхоза (Лисинского УОЛХ). Учебно-опытный лесхоз располагается в Ленинградской области в центральной части Тосненского административного района. Территория лесхоза представляет собой компактный лесной массив, протяженность с севера на юг составляет 34 км, с запада на восток - 18 км. По материалам лесоустройства, которое проводилось в 2005 г., общая площадь составляет 28384 га.

Комплектация применяемой для аэрофотосъемки многороторной платформы (Рис. 1) следующая: рама Tarot FY650 с размером по диагонали 650 мм; моторы T-Motor MN4010 с частотой вращения 475 об./вольт; пропеллеры прямого и обратного вращения из карбона размером 15x5.5”, контроллеры управления скоростью вращения моторов T-Motor с опторазвязкой по каналу управления и номинальной нагрузкой 40A; автопилот Pixhawk с прошивкой Arducopter v3.1.5; система глобального позиционирования GPS Ublox Neo-6m; система передачи данных по радиоканалу 900МГц; литий-полимерные аккумуляторные батареи емкостью 5000 мАч 4s1p с номинальной токотдачей 40C; подвес на бесколлекторных моторах прямого привода Tarot T-2D (Рис. 2); фото-видеокамера GoPro Hero 3 Black Edition.

Рис. 1. Многороторная платформа

Рис. 2. Двухосевой подвес на бесколлекторных моторах прямого привода

В состав наземной станции входят портативный компьютер с установленным программным обеспечением Mission Planner и радиомодем для обеспечения связи с беспилотным летательным аппаратом. Программное обеспечение позволяет выполнять настройку БПЛА, создавать полетное задание, управлять летательным аппаратом и получать данные телеметрии во время полета. План полета представлен в табличном виде (табл. 1).

Предварительно были определены опорные точки для географической привязки планово-картографических материалов. Для визуализации опорных точек на снимках применялись листы фанеры окрашенные в белый цвет размером 0,5x0,5м, которые укладывались на открытые участки местности. Измерения координат производились с помощью GPS-приемника по центру листа фанеры, результаты приведены в табл. 2.

Запуск, полет и посадка многороторной платформы выполняется полностью в автоматическом режиме. Оператор беспилотного летательного аппарата осуществляет контроль за выполнением команд, а в случае возникновения нештатной ситуации, переводит управление БПЛА в ручной режим.

Таблица 1

План полетного задания

№ п/п	Наименование команды автопилота	Широта, град	Долгота, град	Относительная высота, м	Дистанциям	Азимутград
1	TAKEOF	0	0	30	0	0
2	WAYPOINT	59.4092187	30.7002851	150	51.4	35
3	WAYPOINT	59.4151479	30.6970074	150	684.9	344
4	WAYPOINT	59.4148428	30.6953723	150	98.5	250
5	WAYPOINT	59.4092417	30.6984691	150	647.0	164
6	WAYPOINT	59.4092647	30.6966530	150	102.8	271
7	WAYPOINT	59.4145378	30.6937373	150	609.1	344
8	WAYPOINT	59.4142328	30.6921023	150	98.5	250
9	WAYPOINT	59.4092876	30.6948369	150	571.2	164
10	LAUNCH	0	0	0	0	0

Таблица 2

Географические координаты опорных точек

№ опорной точки	Долгота, град	Широта, град	Высота, м
1	30.69730	59.40922	54
2	30.69470	59.41065	56
3	30.69451	59.41249	57
4	30.69331	59.41438	56
5	30.69492	59.41487	56
6	30.69647	59.41507	56

Беспилотным летательным аппаратом выполнена аэрофотосъемка 123 квартала Лисинского учебно-опытного лесхоза, в результате получена серия аэрофотоснимков высокого разрешения (Рис. 3).

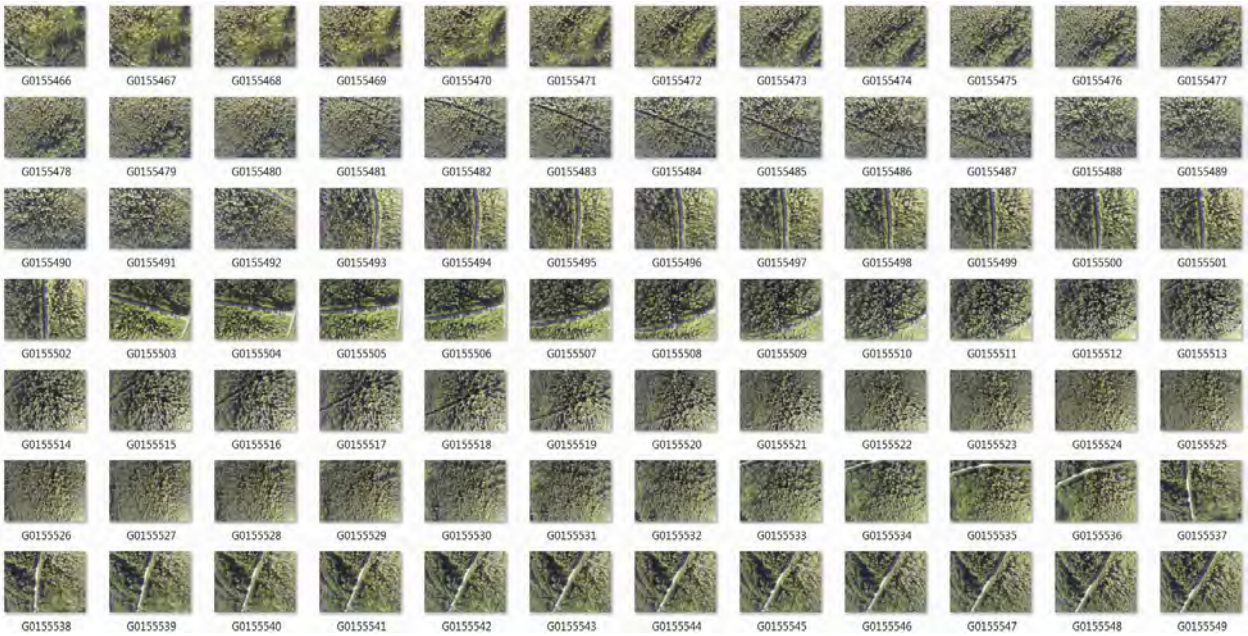


Рис. 3. Серия аэрофотоснимков высокого разрешения на примере Лисинского учебно-опытного лесхоза

Для определения местоположения камеры во время съемки проведен анализ лог-файлов полета. С помощью программы «Трек-Альбом: Летная Книжка» http://track-album.com сопоставлены координаты с центрами снимков. Координаты добавлены в свойства изображений в специальном формате EXIF. EXIF (англ. Exchangeable Image File Format) — стандарт, позволяющий добавлять к изображениям дополнительную информацию (метаданные), комментирующую этот файл, описывающую условия и способы его получения.

Для задания положения и ориентации модели необходимо указать географические координаты минимум для трёх точек. Важно, как можно точнее определить положение опорных точек через их проекции на исходных фотографиях. Для определения положения маркеров в трехмерном пространстве необходимо указать их положение как минимум на двух фотографиях. Чем большее число фотографий используется для указания проекций маркера, тем выше будет точность позиционирования. По результатам обработки получена таблица среднеквадратичных ошибок отклонения положения маркера относительно опорной точки (табл. 3).

Во время процесса выравнивания фотографий фотограмметрическая система PhotoScan оценивает значения параметров внутренней и внешней ориентации камеры, в том числе нелинейных радиальных дисторсий [Никифоров, 2011].

Фотограмметрическая система PhotoScan поддерживает наиболее распространенные типы камер: кадровые камеры, сферические камеры, цилиндрические камеры и камеры с широкоугольным объективом. Для каждого типа камер используется своя модель корректировки дисторсий (Рис. 4).

Таблица 3

Вычисленные среднеквадратичные ошибки положения маркеров относительно опорных точек

№ опорной точки	**Ошибка** X, м	**Ошибка** Y, м
1	6.96789	0.0381546
2	0.705149	3.16084
3	-5.43435	-2.43173
4	0.517607	2.57847
5	-1.96702	-3.13761
6	-2.07492	2.80636
Общая	3.80839	2.59083

По умолчанию PhotoScan рассматривает стандартные для камеры параметры калибровки в качестве первоначального предположения и пересчитывает их во время выравнивания фотографий (Табл. 4).

Используются следующие параметры калибровки камеры:

fx, fy - фокусное расстояние по x- и y- осям (в пикселях);

cx, cy - координаты главной точки, т. е. координаты пересечения оптической оси объектива с плоскостью сенсора;

Skew - коэффициент скоса;

k1, k2, k3 - коэффициенты радиальной дисторсии;

p1, p2 - коэффициенты тангенциальной дисторсии.

Обработка материалов аэрофотосъемки выполнена в специализированной цифровой фотограмметрической системе Agisoft Photoscan [Руководство, 2016]. Серия состоит из 166 изображений с разрешением каждого снимка 3000 x 2250 пикс, пространственное разрешение съёмки составило 7,13 см/пикс.

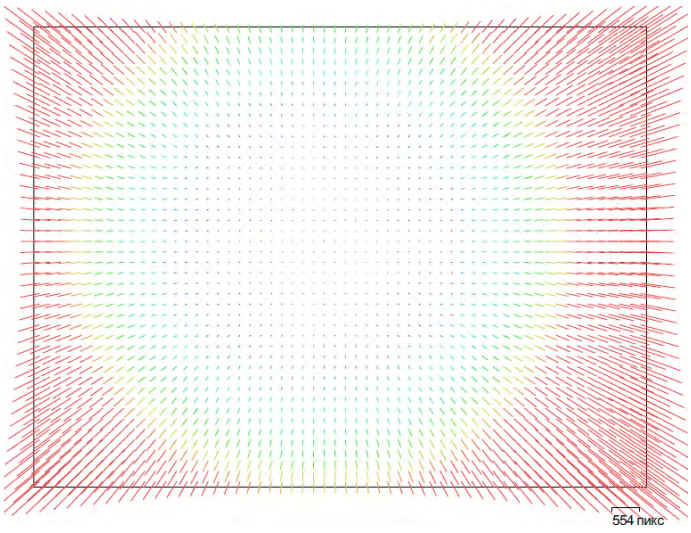


Рис. 4. Невязка по связующим точкам

Таблица 4

Параметры калибровки камеры

Тип камеры	Fx	Fy	Cx	Cy	Skew	K1	K2	K3	P1	P2
Кадровая	1780.5	1781.28	1392.9	1040.81	-1.1047	-0.02874	0.091423	-0.02611	0.001257	-0.00043

После определения положения и ориентации камеры для каждого кадра и создается разреженное облако точек. В результате обработки выполнено выравнивание серии из 166 снимков и построено разреженное облако из 157708 точек.

ЦФС PhotoScan позволяет создавать и отображать плотное облако точек. Основываясь на рассчитанных положениях камер программа вычисляет карты глубины для каждой камеры и на их основе строит плотное облако точек. Плотное облако точек состоит из 16943810 точек.

PhotoScan поддерживает несколько методов восстановления трехмерной полигональной модели и предоставляет ряд настроек, позволяющих выполнить оптимальную реконструкцию для конкретного набора фотографий.

Методы, определяемые типом поверхности «карта высот», оптимизированы для моделирования плоских поверхностей, например, таких как ландшафт. Этот тип поверхности следует выбирать при обработке результатов аэрофотосъемки, поскольку соответствующие методы требуют меньшего количества ресурсов памяти, и следовательно, позволяют обрабатывать большее число изображений. На основе плотного облака точек создана модель, которая содержит 713000 полигонов и 358314 вершин.

Режим наложения текстуры определяет, каким образом текстура объекта будет храниться в текстурном атласе. Выбор подходящего режима помогает получить оптимальный вид хранения текстуры, что ведет к улучшению качества визуализации итоговой модели. Для визуализации использовался общий режим параметризации с размером текстуры 4096 х 4096 пикс.

Карта высот (DEM) представляет модели поверхности в виде регулярной сетки значений высоты. Карта высот может быть рассчитана на основании плотного облака точек, разреженного облака точек или полигональной сетки. Чтобы добиться наиболее точного результата, карта высот создана на основе плотного облака точек с разрешением 4726 х 4910 пикс (Рис. 5).

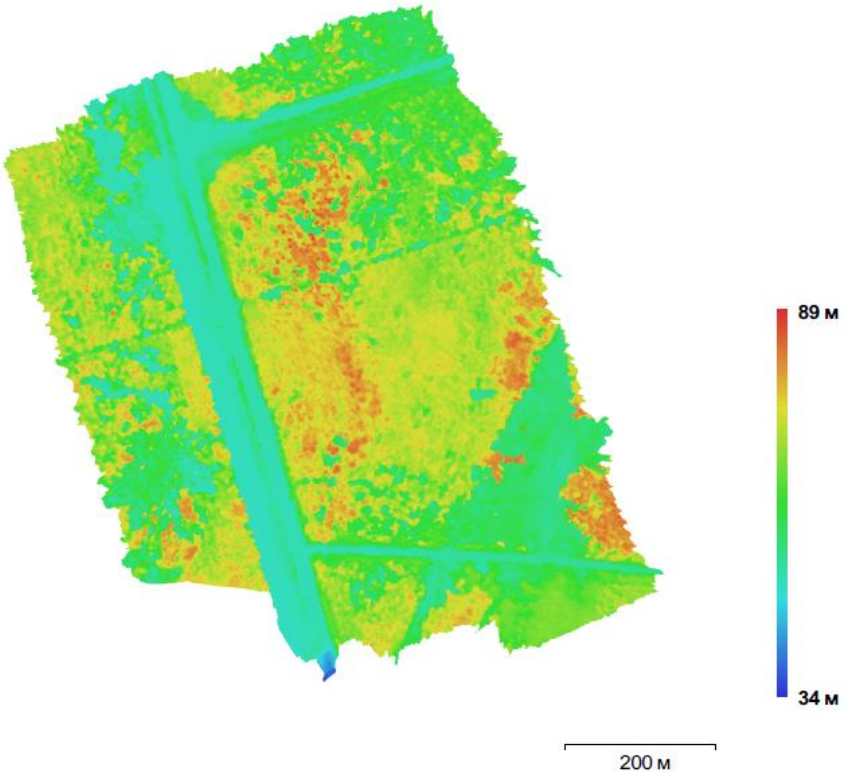


Рис. 5. Карта высот (DEM)

Ортофотоплан строится на основе данных исходных снимков и реконструированной модели, что позволяет создавать результирующее изображение высокого разрешения. Ортофотоплан создан с разрешением 12188 x 13076 пикс. (Рис. 6).

Рис. 6. Ортофотоплан лесного участка 123 квартала Лисинского учебно-опытного лесхоза

Полученный ортофотоплан соответствует требованиям лесоустроительной инструкции [Лесоустроительная, 2011]. На основе ортофотоплана и карты высот возможно выполнить автоматизированное дешифрирование с целью определения границ лесотаксационных выделов и их таксационных характеристик.

В результате проведенных работ, выполнена апробация возможности применения беспилотного летательного аппарата на базе многороторной платформы для аэрофотосъемки участков лесного фонда. Беспилотные летательные аппараты вертолетного типа на основе многороторных платформ занимают свою нишу при выполнении работ, связанных с аэрофотосъемкой. Обладая возможностью вертикального взлета и приземления, они в основном применяются для аэрофотосъемки небольших участков. Для маршрутной аэрофотосъемки значительной протяженности рекомендуется применять БПЛА самолетного типа.

Библиографический список

1. Зинченко, О. Н. Беспилотные летательные аппараты: применение в целях аэрофотосъемки для картографирования [Электронный ресурс]. – Режим доступа:www.url:http://www.racurs.ru/www_download/articles/UAV_1 .pdf - 2011.
2. Лесоустроительная инструкция: приказ МПР России № 516 от 12.12.2011г.
3. Никифоров, А. А. Искажения цифровых фотоснимков высокого разрешения полученных при аэрофотосъемке беспилотными летательными аппаратами [Текст] / А. А. Никифоров // Информационные системы и технологии : сб. науч. тр. - СПб.: СПбГЛТА , 2011. - Вып. 3. - С.17-21.
4. Руководство пользователя Agisoft PhotoScan: Professional Edition, версия 1.2 [Электронный ресурс]. - Режим доступа: www.url: http://www.agisoft.com/pdf/photoscan-pro_1_2_ru.pdf - 10.10.2016.
5. Сухих, В. И. Агрокосмические методы в лесном хозяйстве и ландшафтном строительстве [Текст]: учебник / В. И. Сухих. – Йошкар-Ола: МарГТУ, 2005. - 392 с.
6. А. М. Заяц, А.А. Логачев. Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей //Известия Санкт-Петербургской лесотехнической академии-СПб. : СПбГЛТУ, 2016. № 216, с. 241-255.
7. А. М. Заяц. Беспроводные сенсорные сети в системе мониторинга состояния лесов//сборник докладов международной конференции «Леса России» - СПб. .:СПбГЛТУ , 2016. № 2, с. 23-26

И.А.Обухова, кандидат технических.наук, доцент
Р.М.Яковлев, канд.физ.-мат.наук, ведущий научн.сотр.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ЖИДКОСОЛЕВОГО УРАН-ТОРИЕГО РЕАКТОРА

В прошлых работах [1] и ранее (2015-2016 гг.) сборника кафедры были приведены результаты компьютерных расчетов по радиационной безопасности в околоземном пространстве и в космосе. В данной работе предлагается обсуждение вопросов безопасности применения ядерной энергетики. Широко распространенная урано-плутониевая ядерная энергетика предполагает производство не только энергии, но и компонент ядерного оружия, что противоречит задачам нераспространения ядерного оружия. В рамках урано-плутониевой энергетики возможно решение проблем нераспространения ядерного оружия только путем юридических запретов. В рамках урано-ториевой энергетики на жидкосолевых реакторах решается проблема как производства энергии, так и проблема нераспространения на физико-технологическом уровне, что определяет перспективность уран-ториевой энергетики, которую и называют альтернативной ядерной энергетикой.

Произошло две катастрофы планетарного масштаба и множество ядерных аварий и инцидентов, которые случаются почти еженедельно. Росатом будучи монополистом в области атомной энергетики сохраняет и закрепляет на грядущие десятилетия научно-техническое *отставание* в области естественной безопасности и затратности атомной энергетики. Это объясняется тем, что Россия, в том числе и ведущие государства мира (*программа INPRO 4*), до сих пор используют в ядерных реакторах *твэльные гетерогенные технологии* 50-летней давности, а также взрыво- и пожароопасные теплоносители (вода, натрий), горючие.

Выживание человечества и его благополучие в течение грядущих столетий во многом определится обеспеченностью его энергией, или, как это принято сейчас называть – энергетической безопасностью. Главная составляющая этой безопасности – обеспеченность сырьем. Если исходить из того, что уровень потребления энергии на одного человека в развитых и развивающихся странах постепенно выравнивается, и учесть, что численность населения к 2050 году увеличится в 1,5 раза, то ресурсная база двух основных компонент существующей сейчас энергетики - нефти и газа, будет исчерпана к середине 21 века.

Запасов угля значительно больше и хватит его на несколько сотен лет. Но, к сожалению, широкомасштабное сжигание угля в крупных городах приводит к ужасающему загрязнению атмосферы, что имело место, например, в Лондоне, а в настоящее время происходит в некоторых городах Китая. При добыче угля происходит также загрязнение очень больших территорий, да и отходов после его сжигания непомерно много. Даже учитывая всё возрастающую стоимость нефти и газа, этих ценнейших химиче-

ских продуктов, они все в больших количествах сжигаются в топках тепловых и электрических станций, оставаясь основным ресурсом для большой энергетики. И это несмотря на то, что помимо быстрого сокращения их запасов происходит накопление газов, вызывающих парниковый эффект (двуокиси углерода, метана, оксида азота и галоуглеродов), при котором глобальное изменение климата может стать весьма вероятным.

В сложившейся ситуации сейчас меняется отношение к атомной энергетике, как единственному более чистому, чем уголь, крупномасштабному источнику энергии. Это происходит во многих странах, в том числе и в России. По оценкам МАГАТЭ в течение нескольких следующих десятилетий потребуется построить примерно 2000 АЭС (мощностью в 1000 мегаватт каждая). Это увеличение общей мощности АЭС в пять раз. Для России в соответствии со стратегическим планом предполагается в базовом варианте увеличить общую электрическую мощность до 300 ГВт.

Но атомная энергетика не стала и не может стать лидером в производстве энергии, не исключив саму возможность глобальной катастрофы.

Энергия, выделяющаяся при делении атомных ядер в реакторе, является источником высококонцентрированной энергии. Это свойство делает её весьма привлекательной, но и создаёт высокую потенциальную опасность при её использовании.

Авария в Чернобыле была настоящей катастрофой для СССР, во многом ускорившем его распад. Сначала (в первый год после аварии на Чернобыле) площадь загрязнения более 15 Ku/км^2 получала зона жесткого контроля. Она составила 10340 км^2 с проживанием 272,8 тыс.чел. в 3-х республиках. Радионуклидами загрязнено 23 % территории Белоруссии, на которой расположено 3,6 тыс. населенных пунктов. Авария на ЧАЭС стала причиной РЗ значительных территорий Финляндии, Норвегии, Швеции, Австрии, Болгарии (45 тыс. км^2) и Великобритании. Экспозиция цезия-137 в загрязненных областях превышала 37 кБк/м^2.

Рис.1.Основные очаги радиоактивного загрязнения Европейской части СССР

В настоящее время в качестве внешних границ загрязнения территория при уровне 1 Ku/км^2 по цезию -137. Это 3,2 % всей территории в европейской части бывшего СССР.

В России площадь таких угодий составляет 2,8 млн.га. Больше всего загрязнены Брянская, Калужская, Тульская области. Больше 15 Ku/км^2 только в Брянской области.

Слишком велики были потери и страх не заглушить уверениями очень-очень низкой (10^{-7}) вероятности повтора Чернобыля. Да, действительно, АЭС последних разработок более безопасны и надёжны, чем станции прошлых поколений. Тем не менее при повреждении системы циркуляции теплоносителя (для охлаждения АЭС 1000 МВт требуется 18 куб метров в секунду) при высоком удельном выделении энергии неминуемо возникнет через несколько минут и произойдет расплав топлива при температуре 2800^0С, из которого испарятся почти все вещества, в том числе и радиоактивные изотопы с температурой кипения ниже 2800^0С. При дополнительном разрушении стенки реактора это приведет к выбросу их в атмосферу, загрязняя, как и в случае Чернобыля, радиоактивностью огромные площади, делая их непригодными для жизни. Поскольку все реакторы РБМК типа Чернобыльского лишены, в отличие от реакторов другого типа, толстых стальных сосудов, в которых находится активная зона, то вероятность вылета за пределы реактора радиоактивных продуктов является значительно более высокой. Но и для реакторов со стальной дополнительной оболочкой при расплаве активной зоны неминуем выброс за пределы АЭС очень высокой радиоактивности, что произошло на АЭС Фукусима-1. Хотя реакторы были остановлены и остаточное тепловыделение в тепловыделяющих элементах (ТВЭЛах) было в 100 раз меньше, чем для работающего реактора, этого оказалось достаточным, чтобы сначала оболочки твэлов при отсутствии теплосъёма нагрелись до температуры свыше 1100^0 С, при которой вода превратившись в пар очень энергично взаимодействует с циркониевой оболочкой ТВЭЛов (пароциркониевая реакция $Zr + 2\ H_2\ 0 = 2H_2 + Zr$) с образованием водорода и выделением дополнительного тепла, что приводит с продолжающимся остаточным тепловыделением к повышению температуры ТВЭЛов до $2800\ ^0$С и неминуемому их расплавлению с вылетом из них накопившихся во время работы реактора радиоактивных элементов.

Атомная ***энергетика нуждается в коренной реконструкции. Существующая администрация Росатома, ведомственные НИИ и Проектные институты расписались в своей административной и научной немощи, пойдя на дополнительное штатное допущение расплава активной зоны АЭС (раньше предполагалась возможность пассивного охлаждения активной зоны без возможности расплава) и последующего сбора кориума в контейнмент под корпусом АЭС во время ядерной катастрофы…***

Возможность повторения катастроф типа Чернобыльской и Фукусимской с колоссальными материальными потерями и трагедией переоб-

лучения многих людей на огромных территориях для большой части населения и выселением сотен тысяч жителей с родной земли привело к тому, что во многих странах дальнейшее развитие очень опасной энергетики не только остановилось, но как в Японии, Германии, Швейцарии и др., привело к решению прекратить дальнейшее использование атомной энергетики.В Японии отказались от использования ядерной энергетики, хотя выброс на Фукусиме был примерно в 10 раз меньше по величине, чем в Чернобыле.

Остаточное высокое тепловыделение в отработанном ядерном топливе (ОЯТ) и его высокая радиоактивность являлись причиной многих серьёзных аварий с выбросом большого количества радиоактивных продуктов. Из-за присутствия в нём америция-241, изотопов плутония и других трансурановых элементов сотни тысячи тонн отходов ядерного топлива (ОЯТ) необходимо охлаждать в специальных хранилищах обеспечивая их постоянный отвод тепла несколько тысяч лет.

Авария на Фукусиме драматически продемонстрировала, насколько атомные станции уязвимы в случае поломок систем охлаждения, и это может «пробудить интерес террористов» к атакам на такие станции или же на поломку системы охлаждения в хранилище ОЯТ, что осуществить значительно проще.

Кроме непосредственного очень высокого уровня риска аварии с выбросом огромного количества радиоактивности, атомная энергетика в существующем варианте уран-плутониевого цикла приводит к неконтролируемому распространению ядерного оружия. Типичный легководный реактор электрической мощностью 1000 МВт производит ежегодно ~200 кг плутония. Скорость его накопления составляет в настоящее время 70 т/год. В 90-е годы в мире ежегодно из облученного топлива выделялось 5-22 т плутония, при этом 2-8 т плутония использовалось в легководных реакторах и бридерах. К 2005 году образовался запас энергетического плутония более 200 т.

Планировалось в конце 60-х начале 70-х годов прошлого века производить более 50% электроэнергии и 30% тепловой на АТЭС к началу 21 века. Атомная энергетика, действительно, весьма бурно развивалась, особенно в передовых странах, но бедных запасами газа и нефти: Франции, Японии, Великобритании, Германии, Швеции, а также в США и СССР, что сопряжено было для этих стран с уже развитыми ядерными технологиями для ВПК. На всех АЭС предполагалось производить в самом начале третьего тысячелетия 3-4 тысяч ГВт электроэнергии. Это было зафиксировано на Женевской конференции в 1971 году. Причём, 30-40% должно было производиться в реакторах на быстрых нейтронах. Но так не получилось. Сейчас общая установленная мощность всех АЭС около 370 ГВт(Эл.), а из энергетических реакторов на быстрых нейтронах работает сейчас только один - БН-800 в России (Суперфеникс во Франции и Мондзю в Японии остановлены из-за нескольких опасных ситуаций, которые возникали при

их эксплуатации). Резкое замедление роста числа АЭС обусловлено тем, что атомная энергетика, вопреки ожиданиям, оказалась :

1- весьма опасной из-за возможности крупных аварий (катастроф) с огромным экологическим и экономическим ущербом;

2-являющейся источником распространения основной составляющей ядерного оружия - плутония;

3- весьма неэкономной в использовании ядерного топлива (более 99% урана сейчас идёт в отходы) и, как следствие, в случае ускоренного развития не обеспеченной топливными ресурсами.

Мы провели лингво-комбинаторное моделирование (ЛКМ), которое опирается на наборы ключевых слов, описывающих ту или иную структуру, по которой строятся системы эквивалентных уравнений с произвольными коэффициентами. Эти произвольные коэффициенты задают структурированную неопределенность, которая может быть использована для подстройки модели.

После проведения детального анализа при использовании лингво-комбинаторного моделирования всех существующих реакторных систем, и возможности возникновения в них критических ситуаций, приводящих к разрушению реактора (в том числе и в случае террористических актов и диверсий) мы пришли к выводу, что обозначенные выше проблемы решаются только при переходе на уран-ториевый цикл с реакторами на топливе, находящемся в жидком состоянии.

Экспериментальный реактор такого типа (MSRE) был запущен в 65 году прошлого века и успешно проработал в течение 4 лет в ORNL (*США*). Схема реактора представлена на рис.2.

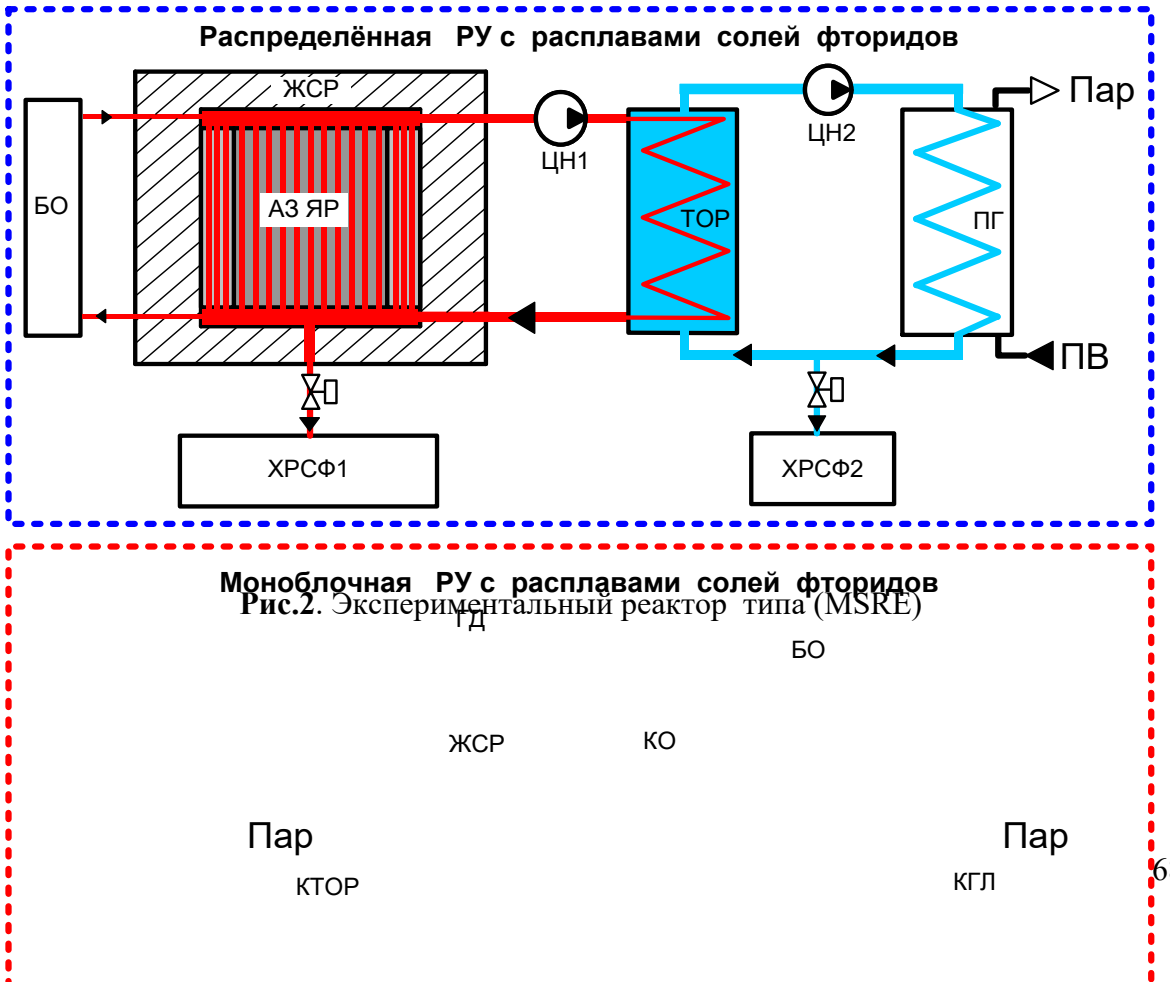


Рис.2. Экспериментальный реактор типа (MSRE)

Принятые сокращения:

АЗ ЯР активная зона ядерного реактора		ПВ питательная вода
БО блок очистки от продуктов ядерных реакций		ПГ парогенератор
ГД газодувка		ХРСФ1 хранилище расплава солей фторидов 1 контура
ЖСР реактор с расплавами солей фторидов в контурах		ХРСФ2 хранилище расплава солей фторидов промконтура
КО компенсатор объёма расплава солей 1 контура		ЦН1 циркуляционный насос расплава 1 контура
КГЛ канал газлифта расплава солей 1 контура		ЦН2 циркуляционный насос расплава промконтура
КТОР каналы теплообмена между расплавами контуров		

На этом реакторе был осуществлён за время работы широкий круг исследований и получены обнадёживающие результаты. Результаты опубликованы в отчётах лаборатории, в научных журналах и докладывались на конференциях. Но затем, работа на этом реакторе была приостановлена. Процесс освоения ториевого цикла в варианте реакторов без твёрдого топлива свёлся к бумажному творчеству. В разных странах были созданы сложные и интересные по весьма широкому кругу ещё и сейчас не решаемых проблем проекты жидкосолевых реакторов большой мощности, причем с довольно высоким коэффициентом воспроизводства. В эти проекты была включена сложнейшая дистанционная переработка циркулирующего ядерного топлива, для некоторых предполагалось дополнительное использование ускорителя (электроядерный бридинг). Эти проекты были грандиозны, но сомнительны в осуществлении из-за их сложности и дороговизны, и поэтому не получили поддержки и финансирования.

Исходя из убежденности, что предельную безопасность и надежность работы реактора легче всего обеспечить в варианте его предельной простоты, мы взялись (и в этом наше отличие от других) за рассмотрение вариантов реакторов с жидким ядерным топливом более простых и надежных, чем MSRE и предлагавшихся по его образцу вариантов больших реакторов.

Какие проблемы обозначились при эксплуатации первого ЖСР (MSRE) в Ок-Риджской лаборатории?

1. Высокая коррозионная активность солевых расплавов, особенно при температуре свыше 700^0C, ко всем типам сталей. Коррозионная активность солевых расплавов возрастает на порядки при образовании следов воды или ионов водорода в солевом расплаве.

2. Охрупчивание стали из-за наработки в солевом расплаве теллура, а также других элементов, приводящих к разрушению стальных труб.

3. Возможная закупорка труб, через которые циркулирует топливная соль из-за их коррозии, а также осаждения на металлических поверхностях целого ряда благородных и полублагородных металлов (Ag, Pd, In, Te, Cd, Mo, Ga, Ge, Nb, Ru, Rh, Zn), которые не образуют в топливной соли стабильных фторидов и также осаждаются на металлических поверхностях первого контура.

4. Необходимость сложной радиохимии, для обеспечения продолжительной работы.

5. Сравнительно низкая радиационная стойкость применявшегося графита, которая приводила к необходимости его замены через 3-4 года.

6. Ограниченное время работы из-за предельно допустимой концентрации продуктов деления в солевой композиции не превышающей 5%.

7. Как и для реакторов с жидкометаллическим носителем остается риск замерзания теплоносителя в трубной системе теплоотвода первого и второго контура.

Эти проблемы существуют в той или иной мере для всех вариантов предлагаемых жидкосолевых реакторов с циркулирующим ядерным топливом по теплообменному контуру. Мы предлагаем совсем отказаться от циркуляции ядерного топлива через внешний контур и осуществлять вместо этого эффективное перемешивание и теплосъём в самой активной зоне.

На следующем рис.3 (подписи те же, что на рис.2) представлен запатентованный нами вариант реактора без циркуляции топлива за пределами активной зоны. Циркуляция топлива происходит с помощью газлифта, при использовании гелия.

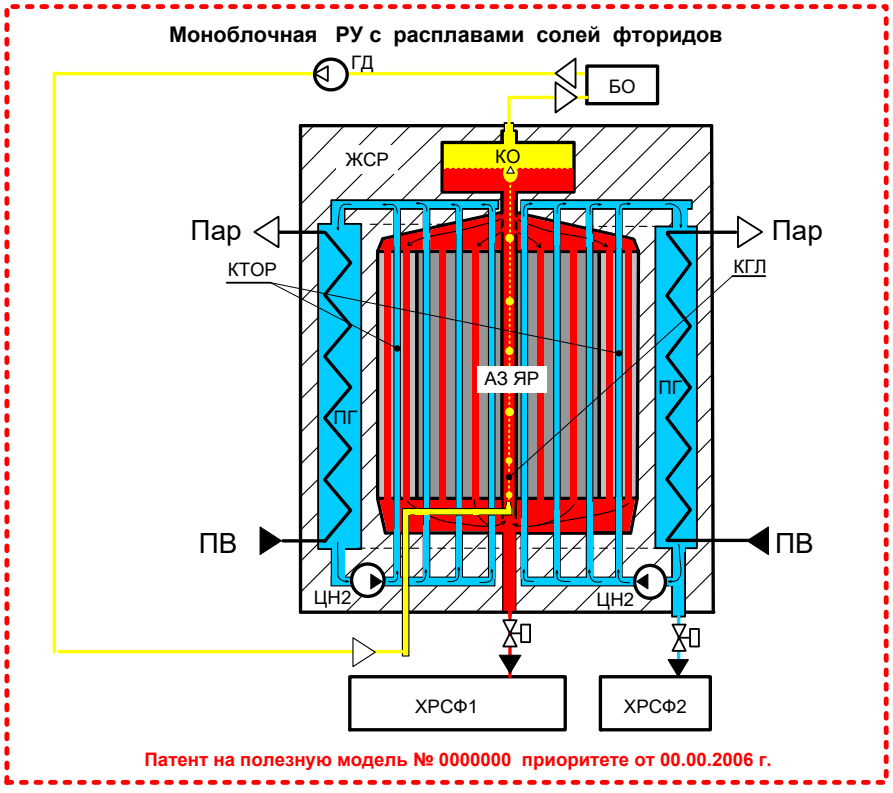


Рис.3. Схема реакторной установки (РУ)

Мы знаем, как осуществить значительно более активное перемешивание и циркуляцию топлива в пределах только активной зоны и тем самым обеспечить высокоэффективный теплообмен с теплоносителем второго контура. даёт возможность использовать в качестве топливной композиции не только расплав соли. Об этом было упомянуто в опубликованной статье «Реактор 2020» (АС, №8, 2006 г.) и патенте на полезное устройство № Патент №64424 [12].

Нужно повысить радиационную стойкость материалов с использованием наноматериалов. Удалить газообразные и летучие радиоактивные продукты предполагается с использованием нанотрубок для эффективного поглощения различных сорбентов при введении в них наноматериалов [4].

Библиографический список.

1. 1.И.А.Обухова,Р.М.Яковлев.Радиационная безопасность в околоземном пространстве и в космосе. Результаты расчетов. Сб.Инф.сист. и технологии:теория и практика. СПбГЛТУ,вып.8.,2016,с.77-86.
2. Р.М.Яковлев, акад. Я.Б.Данилевич, М.Б.Игнатьев, Д.Н.Суглобов. Атомная энергетика без плутония и Чернобыля. Ж. Мир и Согласие №2(35), М. 2008.с.58-64.
3. Я.Б.Данилевич, В.А.Жабрев. Наоматериалы для создания современного энергетического оборудования (на примере турбогенератора 6 МВт, 1200 об/мин). 2-ое Всерос.совещ.ученых, инженеров и произв.в обл.нанотехнологий. М.,15.05.2008.с.18.
4. Р.М.Яковлев, М.Б.Игнатьев и др. Нанотехнологии в альтернативной ядерной энергетике.
5. 2-ое Всерос.совещ.ученых, инженеров и произв.в обл.нанотехнологий. М.,15.05.2008.с.18.
6. Ю.И.Кузякин, Р.М.Яковлев. Транспортная жидкосолевая реакторная установка. Сборник докладов (ч. 1) научно-технической конференции "Корабельная ядерная энергетика – взгляд в XXI век" октябрь 2001 г. Нижний Новгород: ОКБМ.
7. Р.М.Яковлев, М. Б. Игнатьев, Н. Н. Комаров, Ю. И. Кузякин, Д. Н. Суглобов. О реакторах нового поколения. Атомная стратегия XXI, №4, 2005 г.
8. Д.Н. Суглобов, Р.М. Яковлев Времени осталось немного. На сайте ж. Атомная Стратегия XX1, 16/11/2005.
9. Э.Л.Петров, Суглобов Д.Н., Яковлев Р.М. «Реактор-2020», Атомная стратегия XXI, № 24, август 2006 г.
10. Д. Н.Суглобов, Р. М.Яковлев, акад. Б. Ф.Мясоедов. Торий-урановый топливный цикл для тепло- и электроэнергетики, Радиохимия, 2007, т. 49, no. 5
11. D. N.Suglobov, Yu. A.Barbanel, Yu. I.Rodionov and R. M.Yakovlev. Radiochemical Problems of Thorium-Uranium Fuel Cycle Operating as a Molten Salt Reactor, Abstr., 10-th International Seminar on Advanced

Nuclear Fuel Cycle for the XXI Century, 24-27 September, 2007, p. 91, Nizhny Novgorod, Russia.

12. R.M.Yakovlev, Yu.I.Kusyakin, D.N.Suglobov and Y.I.Rodionov. MSR of Average and Low Power with the Lengthened Campaign., Abstr., 10-th International Seminar on Advanced Nuclear Fuel Cycle for the XXI Century, 24-27 September, 2007, p. 94, Nizhny Novgorod, Russia.

13. R.M.Yakovlev, Yu.I.Kusyakin and D.N.Suglobov. Homogeneous Molten Metal Reactor on Fast Neutrons (HMMR) with Dispersed Fuel, Abstr., 10-th International Seminar on Advanced Nuclear Fuel Cycle for the XXI Century, 24-27 September, 2007, p. 94, Nizhny Novgorod, Russia.

14. 13. Ю.И.Кузякин, Р.М.Яковлев. Патент №57040 Ядерная реакторная установка с топливом-теплоносителем в виде расплавов солей фторидов, 27.09.2006.

15. Р.М. Яковлев. Коммерциализация российской науки на английском рынке. Ж. Атомная Стратегия XX1, август 2007.

16. Ю.И.Кузякин, Р.М.Яковлев. Патент №64424 Моноблочная ядерная реакторная установка с жидкометаллическим топливом-теплоносителем, 27.06.2007 г.

17. 16. Р.М.Яковлев, Э.Л.Петров, М.Н.Тихонов, О.Э.Муратов Решение проблем ядерной энергетики в стратегии уран-ториевого топливного цикла. Ж. Атомная Стратегия, май 2007г.

18. Р.М. Яковлев Пять пунктов в пиаровском фантике. Ж. Атомная Стратегия XX1, ноябрь 2007.

19. Р.М.Яковлев, М.Б.Игнатьев, Н.Н.Комаров и др. Лингвинокомбинаторное моделирование наноструктур.2-ое Всерос.совещ.ученых, инженеров и произв.в обл.нанотехнологий. М.,15.05.2008.с.51-52.

20. Р.М.Яковлев, акад. Я.Б.Данилевич, М.Б.Игнатьев, Д.Н.Суглобов. Атомная энергетика без плутония и Чернобыля. Ж. Мир и Согласие №2(35), М. 2008.

21. Д.Н. Суглобов, Р.М.Яковлев. Уран-ториевый цикл и решение проблемы нераспространения ядерного оружия. – Доклад на Международном Пагуошском семинаре №296. С-Петербург, июль 2004.

22. М.Б.Игнатьев, Н.Н.Комаров, Д.Н. Суглобов, Р.М.Яковлев. Чистая атомная энергетика. – Доклад на Юбилейном Пагуошском семинаре, Петербург, ноябрь 2005.

23. Г.М. Воробьёв, Д.Н. Суглобов, Р.М. Яковлев. Дожигание радиоактивных отходов с помощью компактного термоядерного реактора. Межд. Конференция по использованию сферических токамаков. Литва, Вильнюс, 2009 г.

Н.Г. Полетаева, кандидат технических наук, доцент

РЕШЕНИЕ НАУЧНЫХ ЗАДАЧ С ПРИМЕНЕНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Рассмотрим язык программирования Python как инструмент для научных вычислений и анализа данных. Среди многообразия пакетов Python можно выделить следующие библиотеки для этой цели:

- NumPy позволяет удобно работать с векторами и матрицами, при этом реализация всех операций с ними тщательно оптимизирована;
- Matplotlib – для рисования графиков;
- Pandas – для выполнения анализа данных в режиме реального времени;
- Scipy – для решения системы линейных и сложных дифференциальных уравнений, для численного интегрирования и оптимизации;
- SymPy – символьные вычисления.

Эти библиотеки поддерживают все возможности математического пакета Mathcad, но программа, написанная с помощью Matplotlib, может быть встроена в Web-сайт и мгновенно показывать визуализацию данных.

Другие полезные библиотеки:

- MySQLClient – доступ к базам данных MySQL;
- PyODBC позволяет работать с любыми источниками, поддерживаемыми ODBC, с базами данных Access, SQL Server, MySQL и таблицами Excel;
- PyMC – для Байесовского анализа;
- WebBrowser, UrlLib, ftpLib, … – работа с сетью;
- Wget – для специфической работы с Web-сайтом;
- Delorean - для работы с датой и временем;
- Tkinter – графический интерфейс;
- Pillow – работа с изображениями.

Программы Python выполняются интерпретатором, исходные файлы имеют расширение *.py. Основные архитектурные черты Python – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных.

В системе Windows программы Python можно запускать двойным щелчком на файле с расширением .py. При этом происходит запуск интерпретатора и выполнение программы в окне терминала, которое немедленно исчезает после завершения программы. Чтобы избежать этого, можно воспользоваться средой интегрированной разработки. Дистрибутив Anaconda – это лучший выбор, если нужен не только сам интерпретатор Python, но и многие его математические пакеты.

Альтернативные варианты:
- установка Python и его пакетов по отдельности в системе Windows;
- установка Anaconda на виртуальную машину, например, Oracle VirtualBox.

Для иллюстрации применения языка Python и его библиотек рассмотрим задачу построения непараметрических оценок плотности вероятности ядерного типа Розенблата-Парзена:

$$\hat{f}_N(x,\delta) = (N\delta^n)^{-1}\sum_{s=1}^{N} h((x - x^{(s)})/\delta),$$

(1)

где $x^{(s)} \in R_n$ - случайные векторные величины, распределенные с неизвестной плотностью вероятности, $\delta>0$ –параметр сглаживания, $h(\cdot)$ – весовая функция (ядро оценки).

Асимптотические свойства оценки (1) при $N \to \infty$ и $\delta(N) \to 0$ хорошо изучены. Установлены, в частности, условия сходимости поточечной и равномерной по x в различных вероятностных смыслах. Однако все результаты, касающиеся асимптотик оценок, свидетельствуют лишь о принципиальной эффективности оценок и устанавливают те или другие условия на асимптотику убывания параметра усреднения $\delta(N)$.

Асимптотические результаты в принципе не могут дать ответа на центральный вопрос, определяющий практическую работоспособность оценок, как выбрать $\delta(N)$ при заданном конечном числе наблюдений N таким образом, чтобы оценка (1) давала наилучшую (или приемлемую при заданном N) точность. Очевидно, что решение этой задачи может быть получено только при установлении зависимости параметра усреднения от наблюдений, т.е. переходом от детерминированного программно-изменяющегося параметра δ=δ(N) к случайному, зависящему от всего набора наблюдений $\{x^{(s)}\}_1^N$, $\hat{\delta} = \hat{\delta}(\{x^{(s)}\}_1^N)$.

Для выбора параметра сглаживания δ(N) в непараметрических оценках плотности вероятности предложен и исследован метод максимума эмпирического правдоподобия [1].

Суть метода заключается в следующем. Обозначим через $\hat{f}_N^{(j)}(x,\delta)$ оценку плотности вероятности (1), вычисленную по всем узлам $\{x^{(s)}\}_1^N$, кроме j–го:

$$\hat{f}_N^{(j)}(x,\delta) = (N\delta^n)^{-1}\sum_{\substack{s=1 \\ s\neq j}}^{N} h((x - x^{(s)})/\delta). \qquad (2)$$

Эмпирическую функцию правдоподобия определим соотношением:

$$\hat{L}(\delta) = \prod_{j=1}^{N} \hat{f}_N^{(j)}(x^{(j)},\delta). \qquad (3)$$

Здесь каждая из оценок (2) используется для прогноза величины плотности вероятности только в выброшенной точке $x^{(j)}$. В соответствии со смыслом принципа максимума правдоподобия параметр δ выбирается из условия максимума $\hat{L}(\delta)$:

$$\hat{\delta}_N = \arg(\max_{\delta>0} \hat{L}(\delta)). \qquad (4)$$

Если N велико, то все множество узлов $\{x^{(s)}\}_1^N$ можно разбить на два подмножества: обучающее и контрольное, и построить оценку (2) на обучающем множестве, а функцию правдоподобия на контрольном.
Тогда

$$\hat{f}_N(x,\delta) = (N_o\delta^n)^{-1}\sum_{s\in I_o} h((x-x^{(s)})/\delta,$$

$$\hat{L}(\delta) = \prod_{j\in I_k} \hat{f}_N(x^{(j)},\delta), \qquad \hat{\delta}_N = \arg(\max_{\delta>0} \hat{L}(\delta),$$

где I_o, I_k – номера узлов $x^{(s)}$, отнесенных к обучающему и контрольному множествам. Окончательно оценка плотности вероятности определяется формулой $\hat{f}_N(x,\hat{\delta}_N)$.

Задача разбиения экспериментального материала на обучающую и контрольную группы, в данном случае, имеет тот же смысл и порождает те же проблемы, что и в задачах классификации и построения регрессионных зависимостей. Соотношения (2), (3) определяют один из вариантов экономного использования экспериментального материала, при котором одни и те же данные многократно различным образом разбиваются на контрольное и обучающее множества. Таким способом формирования функции правдоподобия $\hat{L}(\delta)$ приходится пользоваться при сравнительно небольших N.

Подчеркнем содержательно наиболее важный момент: то или иное разбиение наблюдений на две группы необходимо для проверки качества оценок – возможности их использования для интерполяции восстанавливаемой зависимости.

Кроме того, при обычных предположениях о свойствах весовых функций *h(·)*, имеет место следующий формальный факт. Если отказаться от разбиения данных на группы и принять $\hat{L}(\delta) = \prod_{j=1}^{N} \hat{f}_N(x^{(j)},\delta)$, то принцип максимума приводит к тривиальному и неинтересному результату $\hat{\delta} = \arg(\max \hat{L}(\delta)) = 0.$

Метод (2) - (4) был реализован в виде вычислительной процедуры на языке программирования Python и опробован построением оценок плотности вероятности $\hat{f}_N(x,\hat{\delta}_N)$ по случайным $\{x^{(s)}\}_1^N$, имеющим заранее заданную плотность вероятности. Расчеты были выполнены с использованием различных весовых функций *h(u)* при различных N и *f(x)*.

Будем характеризовать точность восстановления одномерной плотности вероятности квадратичным функционалом:

$$U_N^2(\delta) = \int_{-\infty}^{\infty} \Delta_N^2(x,\delta)dx \Big/ \int_{-\infty}^{\infty} f^2(x)dx, \qquad (5)$$

$$\Delta_N^2(x,\delta) = M\{(\hat{f}_N(x,\delta) - f(x))^2\}.$$

Назовем оптимальным δ_N^* значение параметра сглаживания, обеспечивающее минимум функционала (5):

$$\delta_N^* = \arg(\min U_N^2(\delta)).$$

Расчеты показывают, что значения $\hat{\delta}_N$, определяемые в силу (4), близки к оптимальным значениям параметра сглаживания δ_N^*.

Исследовались следующие весовые функции:

1) гауссовского типа

$$h(u) = (2\pi)^{-1/2} e^{-u^2/2};$$

(6)

2) финитные типа равномерных

$$h(u) = \begin{cases} 1/2, |u| \le 1 \\ 0, \quad |u| > 1 \end{cases}. \qquad (7)$$

Приведем выражения, определяющие квадратичный функционал $U_N^2(\delta)$ для гауссовского *N(0,1)* и равномерного распределений:

А. *f(x)=N(0,1)*

Для ядра (6)

$$U_N^2(\delta_N) = (N\delta_N)^{-1} + (N-1)/(N\sqrt{1+\delta_N^2}) + 1 - 2^{3/2}/\sqrt{2+\delta_N^2}.$$

(8)

Для ядра (7)

$$U_N^2(\delta_N) = \sqrt{\pi}\{(N\delta_N)^{-1} + (N-1)/(8N\delta_N^2)\int_{-\infty}^{\infty}(erf((x+\delta_N)/\sqrt{2}) - erf((x-\delta_N)/\sqrt{2}))^2 dx -$$

$$2\delta_N^{-1} erf(\delta_N/2)\} + 1,$$

(9)

где $erf(x) = 2/\sqrt{\pi}\int_0^x e^{-u^2} du.$

В. $f(x) = \begin{cases} 1, & x \in [0,1] \\ 0, & x \notin [0,1] \end{cases}$

Для ядра (6)

$$U_N^2(\delta_N) = (2\sqrt{\pi}\, N\delta_N)^{-1} + (N-1)/(4N)\int_{-\infty}^{\infty}(erf(x/(\sqrt{2}\,\delta_N)) - erf((x-1)/(\sqrt{2}\,\delta_N)))^2 dx +$$

$$1 - 2^{3/2}\delta_N(e^{-1/(2\delta_N^2)} - 1)/\sqrt{\pi}.$$

(10)

Для ядра (7)

$$U_N^2(\delta_N) = (2N\delta_N)^{-1} + (N+2)/(3N)\delta_N - 1/N, \quad \delta_N^* = \sqrt{3/2(N+2)^{-1}}.$$

(11)

В табл. 1 и 2 приведены результаты расчетов величин $\delta_N^*, \hat{\delta}_N, U_N^2(\delta_N^*)$ и функционала $\hat{U}_N^2$, который характеризует качество восстановления плотности и определяется формулой:

$$\hat{U}_N^2 = N^{-1}\sum_{j=1}^{N}(\hat{f}_N(x^{(j)}, \hat{\delta}_N) - f(x^{(j)}))^2 / \int_{-\infty}^{\infty} f^2(x)dx.$$

(12)

Таблица 1

f(x)=N(0,1)

N	$h(u) = (2\pi)^{-1/2} e^{-u^2/2}$				$h(u) = \begin{cases} 1/2, \lvert u\rvert \le 1 \\ 0, \quad \lvert u\rvert > 1 \end{cases}$			
	δ_N^*	$\hat{\delta}_N$	$U_N^2(\delta_N^*)$	$\hat{U}_N^2$	δ_N^*	$\hat{\delta}_N$	$U_N^2(\delta_N^*)$	$\hat{U}_N^2$
30	0.58	0.65	0.043	0.010	0.98	1.00	0.013	0.007
100	0.45	0.45	0.019	0.004	0.75	0.75	0.006	0.005
200	0.38	0.38	0.011	0.004	0.65	0.70	0.004	0.004
500	0.32	0.32	0.005	0.002	0.54	0.64	0.002	0.002

Таблица 2

$$f(x) = \begin{cases} 1, & x \in [0,1] \\ 0, & x \notin [0,1] \end{cases}$$

N	$h(u) = (2\pi)^{-1/2} e^{-u^2/2}$				$h(u) = \begin{cases} 1/2, \lvert u\rvert \le 1 \\ 0, \quad \lvert u\rvert > 1 \end{cases}$			
	δ_N^*	$\hat{\delta}_N$	$U_N^2(\delta_N^*)$	$\hat{U}_N^2$	δ_N^*	$\hat{\delta}_N$	$U_N^2(\delta_N^*)$	$\hat{U}_N^2$
30	0.14	0.15	0.105	0.127	0.22	0.27	0.121	0.100
100	0.08	0.06	0.064	0.050	0.12	0.11	0.072	0.050
200	0.05	0.04	0.047	0.040	0.09	0.06	0.053	0.050
500	0.04	0.03	0.031	0.032	0.06	0.06	0.035	0.030

Одним из больших преимуществ пакета Matplotlib является та скорость, с которой можно построить график. В приведенной ниже инструкции импортирован основной модуль библиотеки Matplotlib.pyplot для построения графиков под именем plt, именно так наиболее часто сокращается это длинное имя:

import matplotlib.pyplot as plt,

после этого можно использовать функцию plot(), которая собственно и строит график.

Модуль random позволяет генерировать случайные числа. Прежде чем использовать модуль, необходимо подключить его с помощью инструкции:

from numpy import random.

Например, нормальное распределение генерируется с помощью функции u = random.normal(m, sigma, size=N), где m – среднее значение, sigma – стандартное отклонение, N – размерность массива случайных чисел u.

Функция u=random.uniform(a,b,size=N) генерирует массив равномерно распределенных случайных чисел с плавающей запятой u: $a \le u \le b$.

Приведенная ниже функция density(x, d, N) вычисляет непараметрическую оценку плотности вероятности в $(\cdot)$ x для весовой функции гауссовского типа h(u), где d – оптимальный параметр сглаживания, определяемый по алгоритму (2)-(4) при заданном конечном числе наблюдений N:

```
import math
def h(u):
        return math.exp(-0.5*u**2)/math.sqrt(2*math.pi)

from numpy import random
def density(x, d, N):
   m=0
   sigma=1
   u = random.normal(m, sigma, size=N)
   den=0
   for j in range(1,N+1):
      den=den+h((x-u[j-1])/d)
   den=den/(N*d)
 return den
```

На рис. 1-4 приведены примеры оценок плотностей вероятностей, построенные в силу описанных процедур при использовании различных ядер и определении параметра сглаживания по алгоритму (2)-(4).

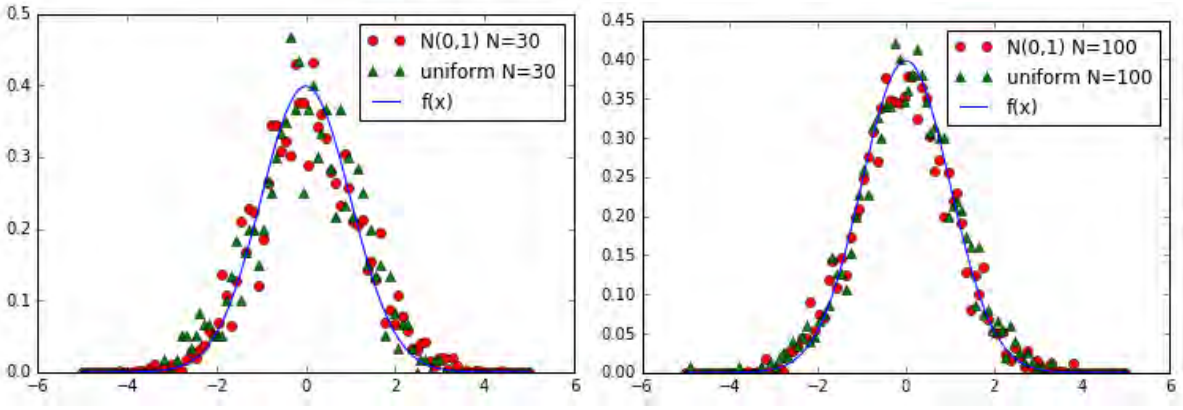


Рис. 1. Парзеновские оценки плотности вероятности *f(x)=N(0,1)*, N=30,100

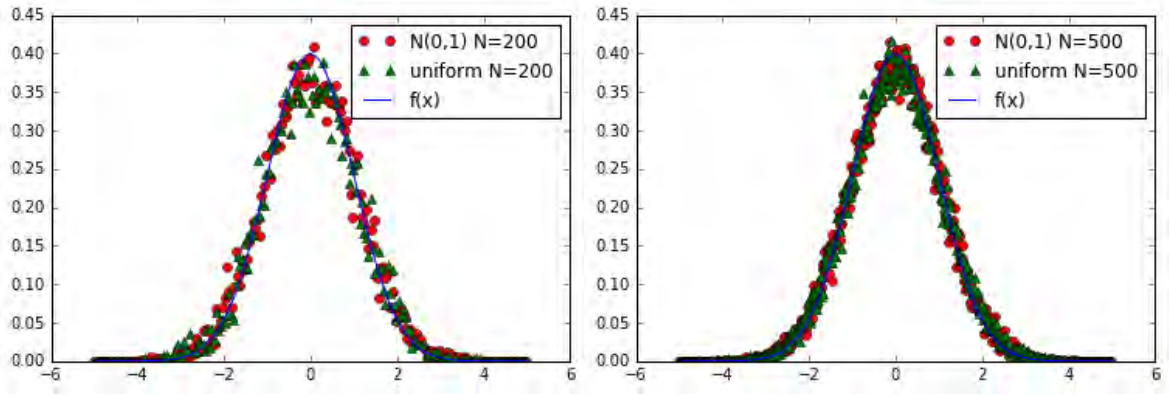


Рис. 2. Парзеновские оценки плотности вероятности *f(x)=N(0,1)*, N=200,500

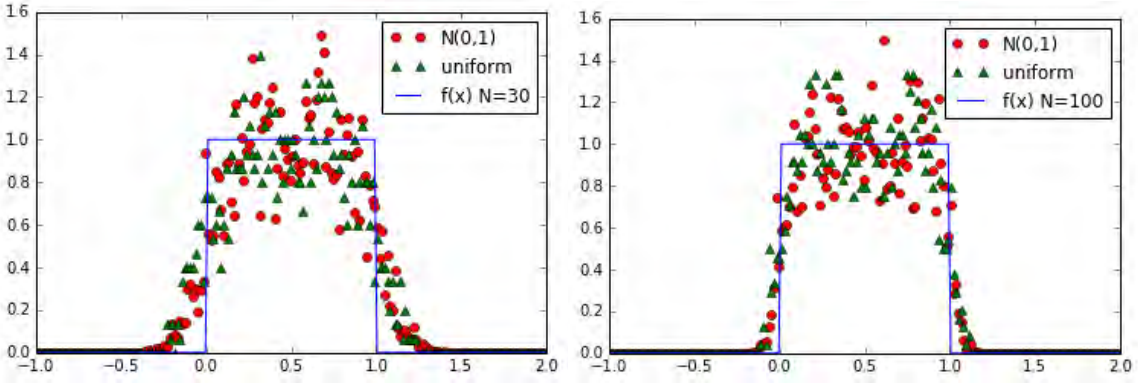


Рис. 3. Парзеновские оценки плотности вероятности равномерного распределения, N=30,100

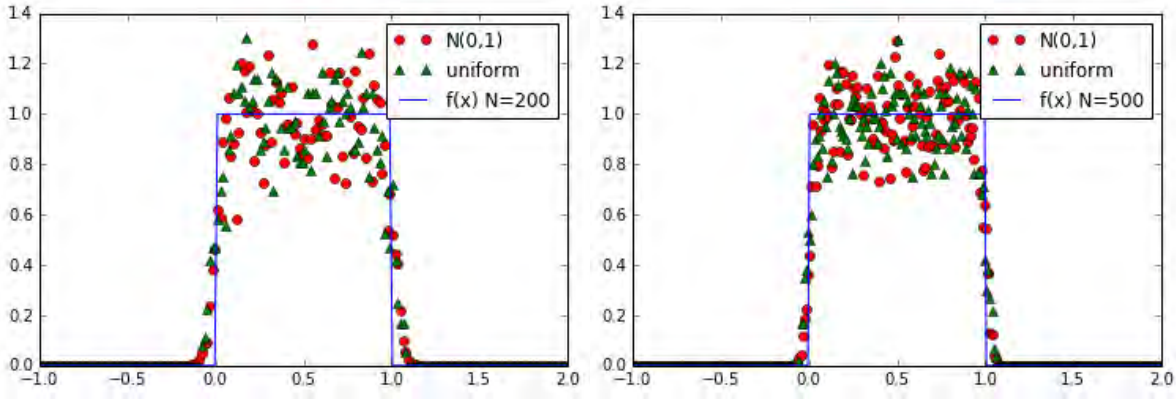


Рис. 4. Парзеновские оценки плотности вероятности равномерного распределения, N=200,500

Приведём простую программу на языке Python, которая строит график Парзеновской оценки нормальной плотности распределения:

```
import numpy as np
import matplotlib.pyplot as plt
N=500
d=0.32   #оптимальный параметр сглаживания для h(u)=N(0,1)
```

```
du=0.64  #оптимальный параметр сглаживания для финитной весовой
#функции типа равномерных, N=500

xs = np.linspace(-5, 5, 500)   # Сетка значений по оси абсцисс

plt.plot(xs, [density(x, d, N) for x in xs], 'ro',
        xs, [densun(x, du, N) for x in xs], 'g^',
        xs, [gaussian(x, 0, 1) for x in xs], 'b-'
       )

plt.legend(['N(0,1) N=500',
          'uniform N=500',
          'f(x)'],              # список легенды
          loc='upper right')    # положение легенды
plt.show()
```

Общие качественные выводы сводятся к следующему:

1. Для неотрицательных весовых функций $h(u) \geq 0$ процедура оказывается весьма эффективной даже при малом N.

2. Разрывные функции *f(x)* восстанавливаются в общем случае существенно хуже непрерывных, дифференцируемых.

3. Наилучшее восстановление плотности вероятности достигается при использовании однотипного с ней ядра.

4. Непрерывные гладкие ядра гауссовского типа приводят к непрерывным гладким оценкам, разрывные ядра типа (7) приводят к разрывным оценкам.

Библиографический список

1. Катковник В.Я. Принцип максимума эмпирического правдоподобия для выбора параметра сглаживания в непараметрических оценках плотности вероятности / В.Я. Катковник, Н.Г. Полетаева. – Вопросы кибернетики. Актуальные задачи адаптивного управления. – М.: ВИНИТИ, 1982, вып. 89, - 164 с., стр. 90-102.

2. Лапко А.В. Непараметрические системы обработки неоднородной информации /А.В. Лапко, В.А. Лапко. – Новосибирск: Наука, 2007. – 197с.

3. Прохоренок Н.А. Python 3. Самое необходимое /Н.А. Прохоренок, В.А. Дронов. – СПб: БХВ-Петербург, 2016. – 464 с.

4. Duin R.P.W. On the choice of smoothing parameters for Parzen estimators of probability density functions. – IEEE Trans. Comput., 1976. – Vol. C-25, № 11, p. 1175-1179.

Л.Г. Пушкарева магистр

ИССЛЕДОВАНИЕ ПРОТОКОЛА WEBSOCKET В ЛАБОРАТОРНОЙ РАБОТЕ НА ПРИМЕРЕ СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ В ГЕТЕРОГЕННЫХ СИСТЕМАХ «WINDOWS –UBUNTU»

WebSocket — это протокол полнодуплексной связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и web-сервером в режиме реального времени.

Технология реализации этого протокола обладает неоспоримыми преимуществами при информационном взаимодействии клиента и сервера в сравнении с протоколом HTTP [1-3].

В статье описывается пример реализация технологии WebSocket в гетерогенных сетях в лабораторной работе, подтверждающий все ее достоинства. Для достоверности тестирования технологии WebSocket в лабораторной локальной сети была сконфигурирована гетерогенная структура, состоящая из локального компьютера с операционной системой Windows 7 и виртуальной машины Oracle VM Virtual Box с Ubuntu 14.04 LTS, между которыми установлено сетевое взаимодействие.

Для работы по протоколу WebSocket была использована свободно распространяемая утилита websocket**d.**

Запуск утилиты в командной строке позволяет подготовить условия, чтобы приложения могли осуществлять взаимодействие по протоколу WebSocket.

Для семейства операционных систем LINUX утилиту websocketd можно скачать с официального сайта *websocketd.com*. (см. рис. 1).

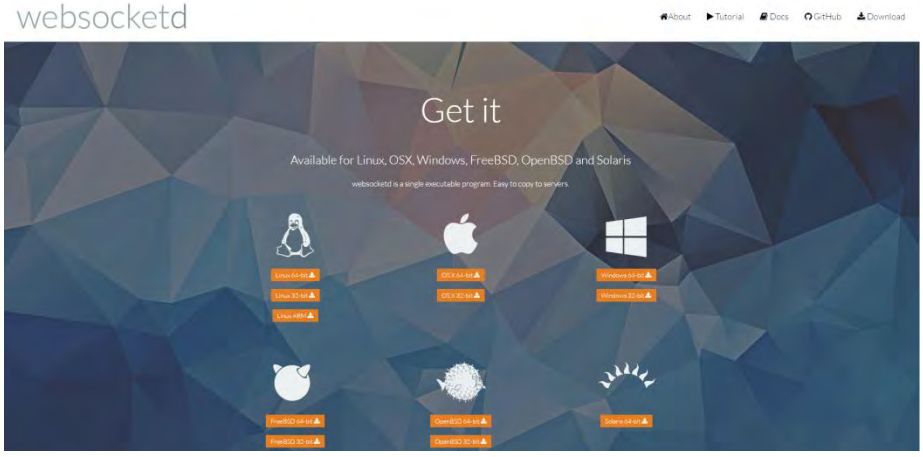


Рис. 1 Официальный сайт утилиты WebSocketD

Для генерирования последовательности сообщений на серверной стороне создадим эхо-сервер, который будет работать следующем образом. Прослушивается определенный TCP или UDP порт, по приходу сообщения на этот порт оно пересылается обратно отправителю, это позволит реализовывать цикл по перенаправлению пользовательских потоков в течение сколь угодно длительного времени и пока соединение активно, пользова-

тель может отправлять сообщения на сервер и получать от него ответ по протоколу websocket.

Для реализации эхо-сервера использован скриптовый язык bash - это стандартная командная оболочка UNIX подобных систем. Скриншот серверного кода на языке bash представлен на рис. 2, в виде файла hello.sh.

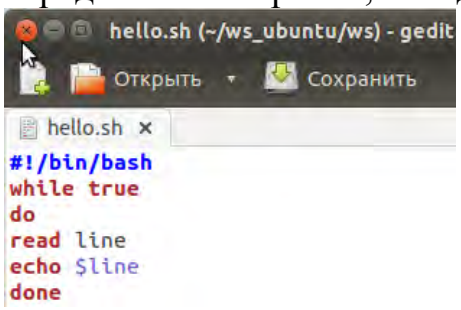


Рис. 2. Скриншот серверного кода на bash

Рассмотрим ситуацию, когда серверная и клиентская части реализованы на виртуальной машине.

Для запуска утилиты websocket**d** на виртуальной машине, необходимо выполнить следующую последовательность действий:

1. Поместить в рабочую папку ранее скачанную с официального сайта утилиту websocket**d**.
2. Создадать (см. рис. 3) командный файл для запуска утилиты websocket**d** совместно с серверным кодом файла hello.sh.

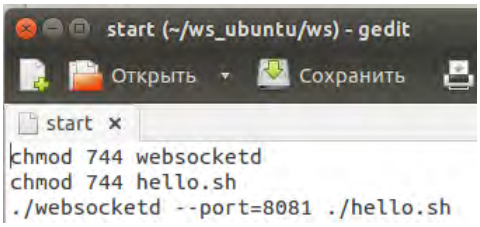


Рис. 3 Скриншот файла для запуска утилиты websocketd

3. Из командной строки виртуальной машины запустить файл на выполнение (см. рис. 4), который с помощью утилиты websocket**d** и приложения hello.sh (на скриншоте выделено желтым цветом), создает WebSocket - сервер соединение по 8081 порту с возможностью осуществлять клиент – серверное информационное взаимодействие.

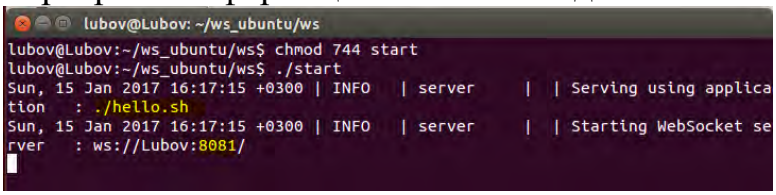


Рис. 4 Скриншот запуска командного файла в терминале

Таким образом, серверная часть приложения запущена и готова к приёму внешних подключений.

Клиентская часть приложения – это запускаемая в браузере html-страница, которая может «общаться» с сервером посредством отправки сообщений по WebSocket-протоколу.

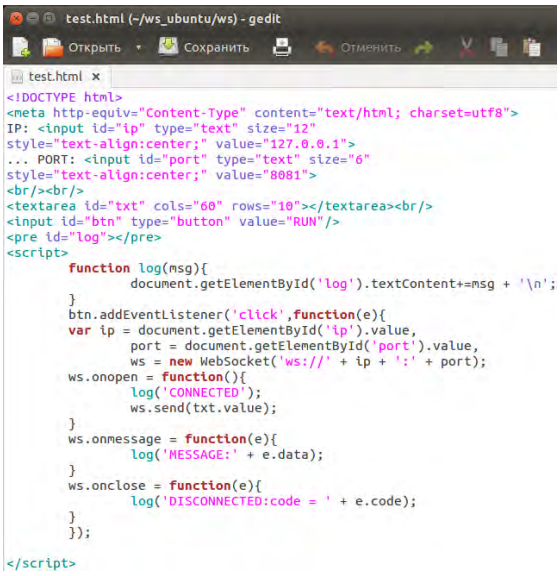


Рис. 5 Скриншот html-кода клиентской части приложения

По умолчанию, сообщения отправляются на сервер по IP адресу 127.0.0.1.

Пользователь имеет возможность изменить значение IP-адреса и подключиться к WebSocket-серверу, запущенному на удаленном компьютере.

Открыв html-страницу в браузере можно проверить взаимодействие клиента и сервера (см. рис. 6).

В терминале, с которого запущено WebSocket - сервер соединение, отображается процесс подключения клиентов (см. рис. 7).

Рис. 6 Скриншот взаимодействия клиента и сервера

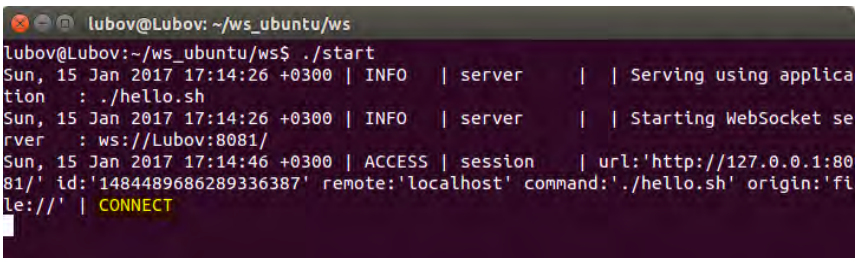


Рис. 7 Скриншот подключения клиента к WebSocket-серверу

Факт подключения клиента к серверу отмечен на скриншоте жёлтым цветом. При этом соединение не разрывается, так как эхо-сервер реализует многократный цикл. Если закрыть браузер текущее соединение разрывается, но при этом сервер продолжает работать.

После запуска и проверки функционирования WebSocket - сервер соединения на виртуальной машине, протестируем эту технологию в гетерогенной системе для двух вариантов:

1. Сервер запущен на Ubuntu, клиент – на Windows 7.
2. Сервер запущен на Windows 7, клиент – на Ubuntu.

Вариант 1. Перенесём клиентскую часть в операционную систему Windows 7 и проведём тестирование. Запустим WebSocket-сервер на Ubuntu.

Затем открываем клиентскую часть в браузере и меняем IP-адрес, установленный по умолчанию, на IP-адрес виртуальной машины с Ubuntu. Отправляем сообщение на сервер (см. рис. 8).

Рис.8 Скриншот взаимодействия клиента Windows 7 и Ubuntu-сервера

В терминале на Ubuntu отображается информация о подключении с IP-адреса 192.168.1.38 (см. рис. 9).

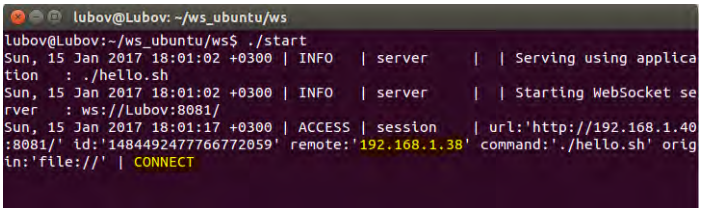


Рис. 9 Скриншот терминала при подключении клиента с Windows 7

На скриншоте информация о подключении выделена жёлтым цветом.

Вариант 2. Для демонстрации работы клиент-серверного приложения в случае, если WebSocket-сервер запущен на Windows 7, как и ранее необходимо скачать с официального сайта версию утилиты websocketd, но для операционных систем семейства Windows.

Эхо-сервер будет реализован средствами языка программирования C# (см. рис. 10).

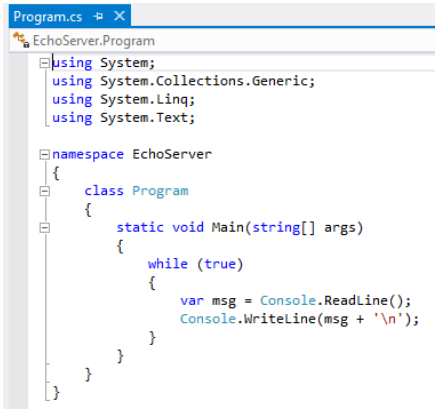


Рис. 10 Скриншот кода эхо-сервера на языке C#

Эхо-сервер на **C#** работает аналогично созданному ранее серверу на языке **bash**.

Создадим командный файл для запуска утилиты websocketd совместно с серверным кодом файла (см. рис. 11) под Windows.

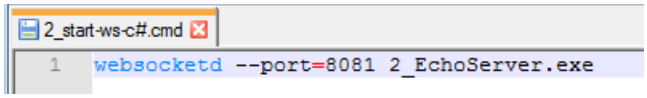


Рис. 11 Скриншот командного файла для запуска утилиты websocketd

Запустим файл в командной строке, WebSocket-сервер организован.

Запустим клиентскую часть приложения в браузере на виртуальной машине с Ubuntu и отправим сообщение на сервер, расположенный на Windows 7 (см. рис. 12).

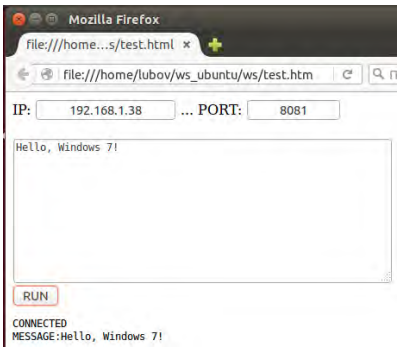


Рис. 12 Скриншот взаимодействия клиента Ubuntu и сервера Windows 7

Тем временем, в командной строке Windows 7 отображается информация о подключении с IP-адреса 192.168.1.40 (см. рис. 13). Информация о подключении отображается на скриншоте желтым цветом.

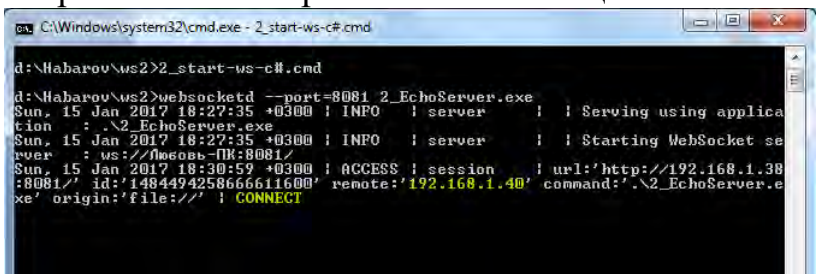


Рис. 13 Скриншот командной строки при подключении клиента с Ubuntu

Проведенное тестирования убедительно показывает, что технология WebSocket успешно работает как в однородных, так и гетерогенных средах, подтверждая все свои преимущества в использовании.

Библиографический список

1. С.П. Хабаров. Взаимодействие узлов сети по протоколу WebSocket. //Информационные системы и технологии: теория и практика: сборник научных трудов № 9 - СПб.: СПбГЛТУ, 2017.
2. С.П. Хабаров. Использование утилиты WebSocketD для удаленного выполнения программ. //Информационные системы и технологии: теория и практика: сборник научных трудов № 9 - СПб.: СПбГЛТУ, 2017.
3. А. М. Заяц, Н.А. Дмитриенко. Подход к организации передачи данных датчиков в информационной системе мониторинга лесных территорий. //Информационные системы и технологии: теория и практика: сборник научных трудов № 9 - СПб.: СПбГЛТУ, 2017.

С.С. Турбин аспирант

ВЫБОР СРЕДСТВ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ВАРИАНТОВ ОРГАНИЗАЦИИ РЕЗЕРВИРОВАННОЙ КОММУНИКАЦИОННОЙ СИСТЕМЫ

При проектировании информационных систем различного прикладного назначения [1,2] требуется обеспечить высокую надежность [3] и устойчивость функционирования [4] как системы, в целом, и в ее коммуникационной подсистемы. Задача обеспечения надежности функционирования особенно остро стоит для систем, реализованных, на основе беспроводных сетей [5-7]. Для обеспечения требуемой надежности, производительности и отказоустойчивости коммуникационной подсистемы используется резервирование коммуникационных средств, в том числе агрегирование каналов при различных вариантах организации надежной доставки

пакетов с использованием механизмов резервированной передачи пакетов их повторных передач и фрагментации [8-12].

Выбор рациональных вариантов структуры сети, организации обмена требует реализации моделирования. Известные аналитические модели поддержки проектирования систем с агрегированием каналов, не учитывают влияние на надежность и эффективность доставки данных множественного доступа, кроме того их возможности ограничены показательным распределением предполагаемых потоков запросов и времени обслуживания. Имитационные модели систем с агрегированием каналов и резервированной передачей пакетов, рассмотренные в [13] – не учитывают реализации множественного доступа и конкретных протоколов обмена.

В связи с этим в настоящей статье ставится задача анализа и выбора средств имитационного моделирования, которые позволяют учесть стандартные протоколы и параметры коммуникационного оборудования, используемого в современных сетях.

Имитационная модель поддержки проектирования резервированных сетевых структур должна удовлетворять все потребности по анализу данных с возможностью масштабирования на более сложные топологии.

При этом имитационная модель должна обеспечивать:

-масштабируемость: возможность запуска моделирования с большим числом вычислительных узлов за разумное время;

-гибкость: возможность указывать в определенном конфигурационном файле соответствующие параметры моделирования;

-соответствие сетевым требованиям: настраиваемая топология сети с реалистичной полосой пропускания, задержки пакетов, должны быть обеспечены потери;

-повторное использование результатов моделирования;

-статистика: сбор статистических данных, в которые входят отправленные, полученные пакеты, пересылка сетевого трафика на узел, успешная или неудачная доставка, число пакетов; предоставленные результаты должны иметь формат, который легко поддается дальнейшему анализу (например, генерация графиков, настраиваемые фильтры и т.д.);

-визуализация: для того, чтобы проверить ошибки в протоколах передачи и топологии сети, необходимо предоставить графический интерфейс для визуализации данных.

На сегодняшний день существует множество различных сетевых имитаторов для построения сетевых моделей. Выбор симулятора очень важен при моделировании и анализе производительности сети. Хороший симулятор это тот, который прост в использовании, обеспечивающий гибкость в разработке модели, позволяющий осуществлять модификацию и проверку, включающий соответствующий анализ выходных данных моделирования, генератор псевдослучайных чисел, а также обладающий статистической точностью результатов исследования.

Среда имитационного моделирования включает в себя интегрированный, универсальный, простой в использовании графический интерфейс,

инструменты, позволяющие проектировщику разрабатывать и моделировать вычислительные сети, такие как *SNMP, TL1, TFTP, FTP, Telnet* и пр.

Существует множество пакетов сетевого имитационного моделирования, таких как *OPNET, NS2, NS3, NetSim, OMNeT++, REAL, J-Sim, QualNet*.

NS2 - дискретная среда моделирования, ориентированная на сетевые события. Обеспечивает поддержку моделирования протокола TCP, протоколы многоадресной передачи (проводные и беспроводные) В данной среде не предусмотрены протоколы маршрутизации.

NS3 - среда сетевого имитационного моделирования для дискретных событий, с открытым исходным кодом. Предназначен в первую очередь для исследовательских и образовательных целей.

OPNET (*Optimized Network Engineering Tools*): - обширный и мощный пакет программного обеспечения для моделирования сетей с широким разнообразием возможностей и поддержкой различных протоколов.

NETSIM (*NetworkBased Environment for Modellingand Simulation*): - приложение, предназначенное для имитации работы сетевого оборудования и программного обеспечения производителя CiscoSystems и предназначен для оказания помощи пользователю в изучении командной структуры *Cisco IOS*.

JSIM (*Java-based simulation*): - система для построения количественных цифровых моделей и анализа их в отношении экспериментальных справочных данных.

OMNet++: - расширяемая, модульная библиотека на основе компонентов *C++* и фреймворк, в первую очередь для построения сетевых симуляторов. Модель строится из иерархически вложенных модулей, которые обмениваются данными с помощью передачи сообщений. Для определения структуры модели *OMNet++* использует язык топологии, *NED*. Рассматриваемая система моделирования включает в себя простой в использовании графический интерфейс, содержащий все необходимые возможности для проектировщика сети.

OMNeT++ реализует множество функций, необходимых для визуализации. Встроенный графический интерфейс отображает топологию сети, узлы и передаваемые сообщения. Возможности отладки позволяют проводить детальный анализ содержимого информационных пакетов и параметров узлов. Графический интерфейс может быть отключен для более быстрого моделирования. На рисунке 1 представлен пример графического исполнения имитационной модели, состоящей из нескольких вычислительных узлов.

Явное преимущество *OMNet++* по отношению к другим решениям - это использование фреймворка *INET*. Данный фреймворк содержит многочисленные модели сетевых устройств и протоколов, включая модели *MAC Ethernet*, поддержку дополнительных механизмов организации очередей, например, с помеченным приоритетом. Это помогает уменьшить время разработки модели. Пакет состоит из библиотек классов языка про-

граммирования *C++* для ядра системы, а также вспомогательных классов (для генерации случайных чисел, сбора статистики, разработки топологии и пр.). Эти классы используются для создания компонентов имитационной модели: модули, каналы и пр.

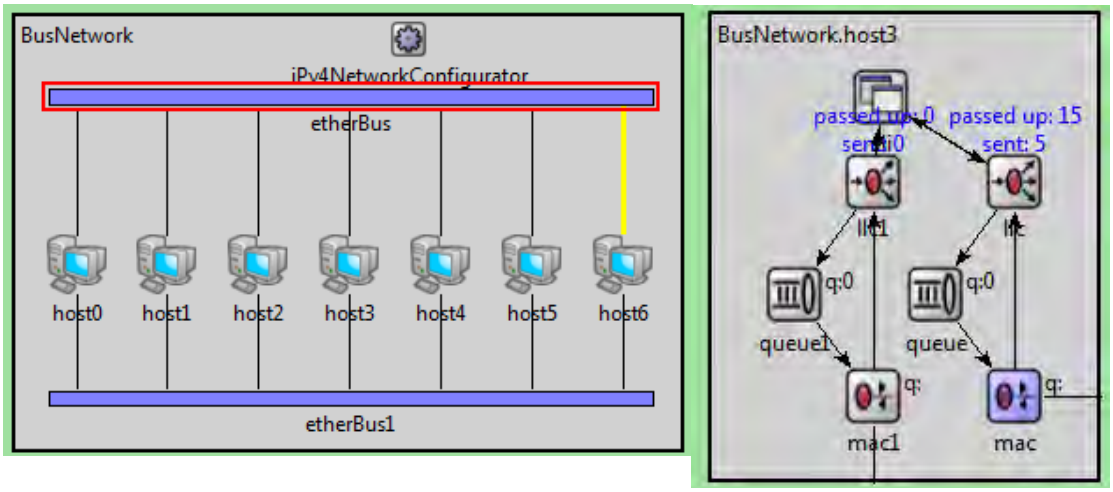


Рис.1. Графический режим в пакете Omnet++

Выбор подходящего пакета моделирования для конкретного приложения не является легкой задачей. Для выбора подходящей программы, важно иметь знания о инструментах, доступных наряду с анализом сильных и слабых сторон. Важно также гарантировать, что результаты, полученные с помощью симуляторов действительны и заслуживают доверия. Был рассмотрен целый ряд широко используемых сетевых симуляторов, в том числе *NS-2* и *OPNET Modeler*. *NS-2* является популярным сетевым симулятором среди научного сообщества, который доступен для скачивания без каких-либо затрат. Тем не менее, *NS-2* трудно использовать, пакетимеет скудную документацию. *OPNET* является коммерческим пакетом, который имеет обширную библиотеку моделей, удобный графический пользовательский интерфейс (*GUI*), и настраиваемые презентации результатов моделирования. Тем не менее, *OPNET* дорогой пакет, даже если он будет использоваться в рамках академических программ университета. Для решения поставленной задачи была выбрана среда имитационного моделирования *OMNet++*. Мотивацией использования *OMNet++* в качестве пакета моделирования для исследований является то, что он предлагает объединенные преимущества *NS-2* и *OPNET*, является бесплатной средой с открытым исходным кодом.

Таким образом, проведенные исследования показали целесообразность использования при построении моделей поддержки проектирования резервированных инфокоммуникационных систем средств имитационного моделирования *OpnetModeler, OMNet++* с расширенными библиотеками моделирования.

Библиографический список

1. Заяц А.М., Логачев А.А. Математические модели для поддержки принятия решений по предупреждению лесных пожаров при ограниченном объеме исходных данных.//Известия высших учебных заведений. Приборостроение. 2016. Т. 59. № 5. С. 342-347.

2. Aleksanin S.A., Zharinov I.O., Korobeynikov A.G., Perezyabov O.A., Zharinov O.O Evaluation of chromaticity coordinate shifts for visually perceived image in terms of exposure to external illuminance// ARPN Journal of Engineering and Applied Sciences. 2015. Т. 10. № 17. С. 7494-7501.

3. Богатырев, В. А. Информационные системы и технологии. Теория надежности: учебное пособие. — М. : Издательство Юрайт, 2016. — 318.

4. Богатырев В.А., Богатырев С.В. Критерии оптимальности многоустойчивых отказоустойчивых компьютерных систем // Научно-технический вестник информационных технологий, механики и оптики - 2009. - № 5(63). - С. 92-98.

5. Заяц А.М Беспроводные сенсорные сети в системе мониторинга состояния лесов//Леса России: политика, промышленность, наука, образование материалы научно-технической конференции. Под. ред. В.М. Гедьо. 2016. С. 154-156.

6. Заяц А.М, ЛогачевА.А. Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей. Известия Санкт-Петербургской лесотехнической академии. 2016. № 216. С. 241-254.

7. Шубина М.А.Использование беспилотных летательных аппаратов для аэрофотосъемки в целях картографирования наземных объектов. Информационные системы и технологии: теория и практика//Сборник научных трудов. отв. редактор А. М. Заяц. 2015,Изд-во: Санкт-Петербургский государственный лесотехнический университет им. С.М. Кирова(Санкт-Петербург) с. 64 -79.

8. Богатырев В.А. Богатырев А.В. Модель резервированного обслуживания запросов реального времени в компьютерном кластере //Информационные технологии 2016. N 5, Т 22, С. 348—355.

9. Богатырев А.В., Богатырев В.А. Надежность функционирования кластерных систем реального времени с фрагментацией и резервированным обслуживанием запросов // Информационные технологии - 2016. - Т. 22. - № 6. - С. 409-416.

10. Bogatyrev V.A. Protocols for dynamic distribution of requests through a bus with variable logic ring for reception authority transfer Automatic Control and Computer Sciences, vol. 33, No. 1, 1999, pp. 57-63.

11. Bogatyrev V.A. An interval signal method of dynamic interrupt handling with load balancing Automatic Control and Computer Sciences, vol. 34, No. 6, 2000, pp. 51-57.

12. Богатырев В.А. Комбинаторный метод оценки отказоустойчивости многомагистрального канала.//Методы менеджмента качества. 2000. № 4. С. 30-35.

13. Богатырев В.А., Богатырев С.В. Резервированная передача данных через агрегированные каналы в сети реального времени // Известия высших учебных заведений. Приборостроение - 2016. - Т. 59. - № 9. - С. 735-740.

14. Попцова Н.А., Богатырев В.А., Кармановский Н.С., Паршутина С.А., Воронина Д.А., Богатырев С.В. Имитационная модель поддержки проектирования инфокоммуникационных резервированных систем // Научно-технический вестник информационных технологий, механики и оптики - 2016. - Т. 16. - № 5(105). - С. 831-838.

15. Дискретный симулятор событий OMNet++ [Электронный ресурс]: официальный сайт проекта. URL: https://omnetpp.org/.

С.П. Хабаров, кандидат технических наук, доцент

ИСПОЛЬЗОВАНИЕ УТИЛИТЫ WEBSOCKETD ДЛЯ УДАЛЕННОГО ВЫПОЛНЕНИЯ ПРОГРАММ

Протокол WebSocket поддерживает технологию двунаправленной связи между браузером и сервером. При подключении к серверу браузер посылает ему HTTP заголовок с запросом о его поддержки WebSocket. Если сервер отвечает положительно, то дальше HTTP прекращается и общение идёт по протоколу WebSocket, который не имеет с HTTP ничего общего и позволяет обеим сторонам обмениваться данными практически в реальном времени и почти без лишнего сетевого трафика.

WebSocket — это самое кардинальное расширение протокола HTTP с его появления. Изначально синхронный протокол на модели «запрос — ответ» становится полностью асинхронным и симметричным, что дает полную свободу в обмене данными. Пользователь сам решает, как это использовать.

В частности, этот протокол используют для параллельных вычислений, построения распределенных систем и Web-приложений. Составной частью этих задач является удаленное выполнение программ. Целью данной статьи является рассмотрение одного из возможных подходов к ее решению на базе протокола WebSocket.

Подход к реализации WebSocket на клиенте и сервере.

Что касается серверной реализации WebSocket, то здесь имеется широкий выбор разных программных продуктов и подходов. От достаточно мощных серверов приложений до собственных разработок. Серверами приложений, у которых есть поддержка WebSocket, являются:

- GlassFish 4, который является эталонной реализацией платформы Java EE 7 (https://glassfish.java.net/),

• IBM WebSphere Application Server (http://www-03.ibm.com/software/products/ru/appserv-was),
• Apache Tomcat 7 (https://tomcat.apache.org/) и ряд других.

Известны реализации WebSocket серверов на Java, на C#, на Node.js, на асинхронном фреймворке phpDaemon. Однако все они требуют достаточно сложной настройки, а также знания языковых средств, с помощью которых можно ввести серверную обработку.

От этого во многом свободна достаточно простая утилита WebSocketD (http://websocketd.com/), которая запускается в фоновом режиме (демон), организуя серверный сокет. Она имеет реализации под разные типы ОС. Ее архив содержит всего один выполняемый файл, который не требует никакой начальной установки.

Ее достоинством при реализации небольших проектов и знакомстве с технологией работы с протоколом WebSocket, является то, что она позволяет для разработки серверных кодов из множества языков выбрать тот, который более всего доступен разработчику. Серверные коды могут разрабатываться на разных языках. В том числе и на скриптовых, которые входят в состав поставки самих ОС. Таких, как bash (Linux), VBScript и JavaScript (Windows).

Что касается реализации WebSocket на клиентской стороне, то здесь надо отметить, что WebSocket – это современное средство коммуникации, кросс-доменное, универсальное и безопасное. Он поддерживается большинством современных браузеров: Internet Explorer 10+, Mozilla Firefox 11+, Google Chrome 16+, Opera 12.5+ и Apple Safari 6+.

Язык JavaScript имеет собственное API для работы с WebSocket. Однако оно работает только на клиентской стороне, которая обычно реализуется на основе того или иного браузера. При этом не все браузеры поддерживают работу этого API. Поэтому первое, что надо делать на стороне клиента – это выявить поддержку браузером работы WebSocket API:

```
if('WebSocket' in window) {
   // WebSocket поддерживается. Можно писать свой код
} else {
   // Надо использовать другие методы связи
}
```

Для открытия соединения достаточно создать новый объект WebSocket, указав в нём соответствующий протокол и локатор ресурса:

```
var sock = new WebSocket('ws://server/',['soap','xmpp']);
```

Для получения сведений о новых подключениях, входящих сообщениях или ошибках используются обработчики событий, связанные с соединением. С любым объектом класса WebSocket, в том числе и с объек-

том sock, связан набор событий и методов, которые приведены в таблицах 1 и 2.

После организации соединения по протоколу WebSocket уже нет клиента и сервера с фиксированными ролями, а есть два равноправных участника. Каждый работает сам по себе, и когда надо отправляет данные другому. Отправил — и продолжил свою работу. Вторая сторона ответит, когда сможет или захочет — может не сразу, а может и вообще не ответит.

Таблица 1

События WebSocket

Событие	Обработчик	Описание события
open	sock.onopen	Происходит, когда sock устанавливает соединение.
message	sock.onmessage	Происходит, когда клиент получает данные с сервера.
error	sock.onerror	Происходит, когда есть ошибки в коммуникации.
close	sock.onclose	Это событие происходит, когда соединение закрывается.

Таблица 2

Методы WebSocket

Метод	Описание
sock.send()	Метод send(data) передает данные, используя соединение.
sock.close()	Используется для прекращения любого существующего соединения.

Общий подход к решению задачи

Предлагается Web-технология, которая на сервере использует утилиту WebSocketD совместно с сервером сценариев Windows Script Host (WSH) в качестве инструмента для удаленного выполнения кода на языках VBScript или JScript. При этом клиентом для ввода скриптов и получения результатов их работы может быть любой браузер в любой ОС, который поддерживает протокол WebSocket. Другими словами ставится задача по использованию браузера в качестве интерфейса ручного или файлового ввода VBS- или JS-скриптов с целью передачи их на сервер для выполне-

ния и возврата результатов клиенту. Это требует разработки серверного Web сервиса, а также html страницы, которая обеспечивает подключение клиента к этому сервису.

Предлагаемую технологию рассмотрим на простейшем примере. Пусть требуется удаленно выполнить небольшой скрипт:

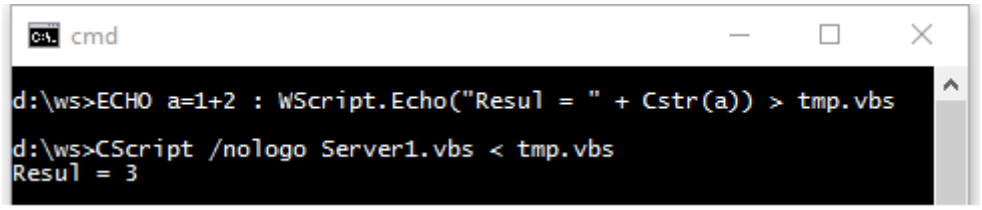
Файл test.vbs

```
For i=1 To 5
   s=s+i
Next
WScript.Echo("Result = " & s)
```

Начнем с реализации серверной процедуры, задача которой принять входящий от клиента поток скриптового кода и передать его WSH на выполнение. Эта задача на VBScript реализуется всего одной строкой:

```
Call Execute(WScript.StdIn.ReadLine())
```

Сохраним скрипт под именем Server1.vbs. Для проверки запустим его в командной строке с помощью сервера сценариев, передав ему два оператора VBScript, которые разделены символом двоеточия.



Как видно из скриншота этот простой серверный код успешно работает с однострочными скриптами. Чтобы его можно было бы использовать в более общем случае, надо чтобы клиентское приложение могло преобразовывать вводимый в нем скрипт, как например test.vbs, в одну строку, передаваемую на сервер по протоколу WebSocket.

В качестве клиентов будем использовать html страницы, для хранения которых на сервере выделим отдельную папку, например, htm. Создадим в этой папке пока пустой файл с именем client_vbs.htm. Успешная проверка локальной работоспособности сервера позволяет сформировать командный файл для запуска Web-сервиса, реализующего разработанный серверный код:

Файл RunServerVB.cmd

```
websocketd --port=8081 --staticdir=.\htm CScript /nologo Server1.vbs
```

После запуска этого файла на выполнение откроется окно командной строки, из которого видно, что сервис WebSocketD запущен и готов к соединению с клиентом по порту 8081 (рис. 1)

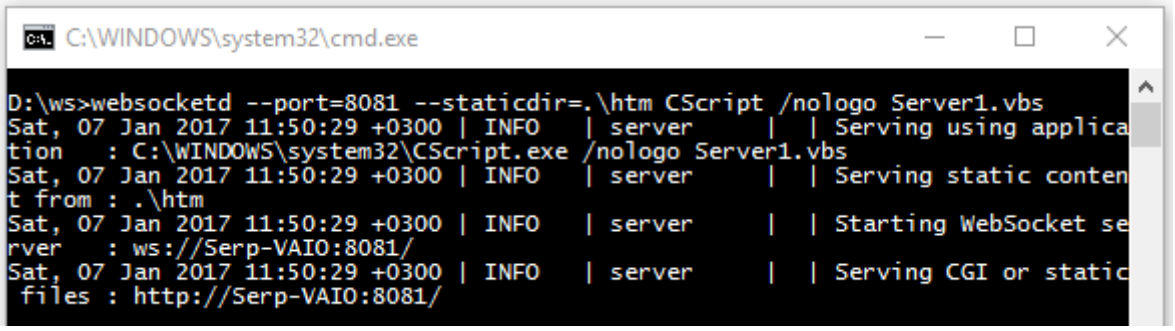


Рис. 1. Запуск Web сервиса на сервере

Проверить работоспособность Web сервиса можно с помощью браузера локального ПК на базе ОС Windows и с любого сетевого ПК, например, с ОС Ubuntu [1,2], зная при этом IP адрес сервера (рис. 2).

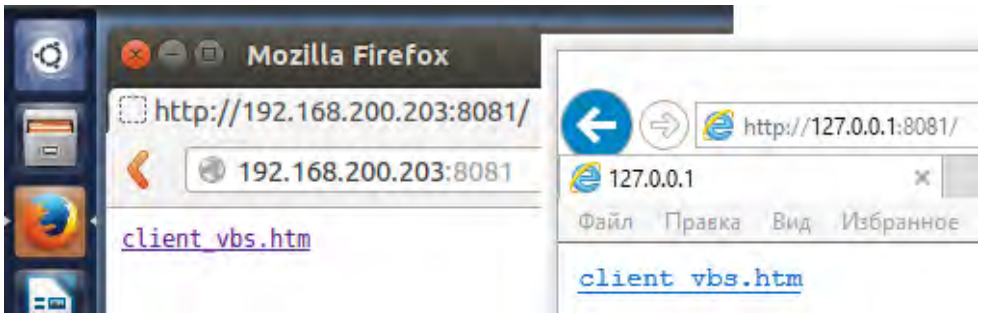


Рис. 2. Доступ к Web сервису с локального и удаленного компьютеров

Убедившись в работоспособности доступа к Web сервису, теперь можно перейти к насыщению его функциональности и разработать необходимую для решения поставленной задачи интерфейсную html страницу, которую клиент будет загружать с сервера. Одним из возможных вариантов может быть представленный ниже код:

Файл client_vbs.htm

```
<!DOCTYPE html>
  <meta http-equiv="Content-Type"
   content="text/html; charset=utf-8">
  <textarea id="txt" cols="60" rows="10"></textarea><br/>
  <input id="btn" type="button" value="R U N"/>
  <pre id="log"></pre>
<script>
if('WebSocket' in window) {
  btn.addEventListener('click', function(e) {
    var ws = new WebSocket('ws://192.168.200.203:8081');
    ws.onopen = function() {
       log('CONNECTED');
       //для VBScript – замена \n на ':'
           ws.send(txt.value.replace(/\n/g, ':')+'\n');
```

```
        //для JavaScript – замена \n на ’’
        //ws.send(txt.value.replace(/\n/g, '')+'\n');
    };
    ws.onmessage = function(e) {
      log('MESSAGE: ' + e.data);
    };
    ws.onclose = function(e) {
      log('DISCONNECTED: code = ' + e.code);
    };
  });
  function log(msg) {
   document.getElementById('log').innerHTML += '<br>'+msg;
  }
} else { alert("WebSocket НЕ поддерживается !!!."); }
</script>
```

После того, как эта страница будет открыта в браузере, в ее текстовое поле будет введен текст VBS скрипта, аналогичный файлу test.vbs, и нажата кнопка RUN, будет получен результат, представленный на рис. 3.

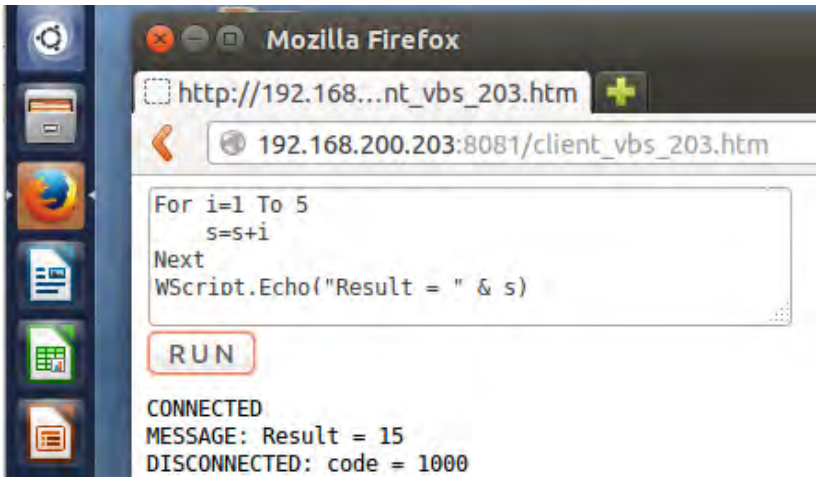


Рис. 3. Выполнение VBS скрипта на компьютере с ОС Ubuntu

Аналогичный Web сервис может быть реализован на компьютере с ОС Windows и для удаленного выполнения скриптов на языке JScript. Для этого надо реализовать однострочный сервер, который, как и выше, будет состоять всего из одной строки, и иметь вид:

Файл Server2.js

```
eval(WScript.StdIn.ReadLine());
```

Для запуска этого серверного кода в виде Web сервиса следует создать командный файл, который будет запускать утилиту WebSocketD, которая будет перенаправлять входной на порт 8082 поток на однострочный сервер. Выбор номера может быть произвольный, но в рассматриваемом примере он отличен от предыдущего, чтобы иметь возможность запускать на одном ПК сразу два Web сервиса.

Файл RunServerJS.cmd

```
websocketd --port=8082 --staticdir=.\htm CScript /nologo Server2.js
```

Интерфейсную html страницу для работы с JS скриптами разместим в той же папке htm. Она будет аналогична файлу client_vbs.htm за исключением трех строк, в которых изменены комментарии и номер порта:

Файл client_js.htm

```
. . .
   var ws = new WebSocket('ws://192.168.200.203:8082');
  ws.onopen = function() {
       log('CONNECTED');
       //для VBScript – замена \n на ':'
           //ws.send(txt.value.replace(/\n/g, ':')+'\n');
       //для JavaScript – замена \n на ''
       ws.send(txt.value.replace(/\n/g, '')+'\n');
    };
. . .
```

Для проверки работоспособности Web сервиса надо на удаленном ПК открыть браузер, в его адресной строке ввести адрес и порт сервера, затем выбрать файл client_js.htm. После загрузки страницы в ее текстовой области можно ввести код на языке JScript (test.js) и нажать кнопку RUN. Если в скрипте не было ошибок, то в окне браузера появится результат выполнения этого скрипта на сервере, переданный на страницу клиента.

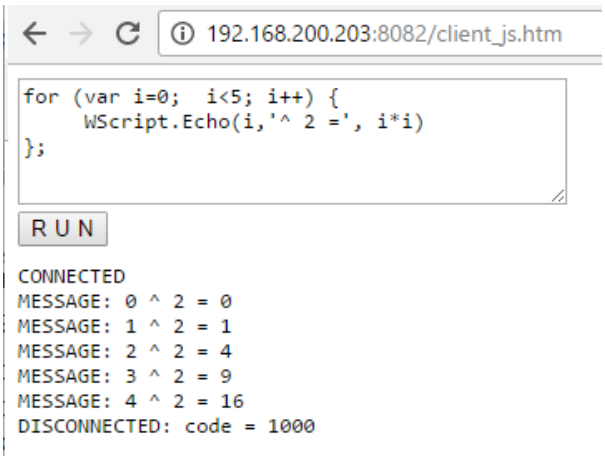


Рис. 4. Ввод и результат выполнение JS скрипта в среде Google Chrome

Как видно из рис. 4 код был успешно передан на сервер, там он отработал под управлением WSH, и пять сообщений было передано в выходной поток. И все эти пять сообщений утилитой websocketd были перенаправлены по протоколу WebSocket к процессу клиента.

Подводя итог этому разделу, можно сказать, что был сформулирован общий подход по использованию WebSocketD для удаленного выполнения скриптовых кодов.

Расширение функциональности клиентской части

На клиентской стороне затруднительно каждый раз в текстовом окне html страницы набирать тот или иной скрипт или копировать его из какого-либо файла. Значительно проще было бы загружать код непосредственно из файла. Для этого можно использовать введенный в HTML5 элемент формы <input type="file" >, который позволяет выбрать нужный файл из стандартного окна, аналогичного FileOpen в Windows.

Есть некоторые особенности использования этого компонента в разных браузерах, но на этом мы останавливаться не будем, а постараемся решить поставленную задачу, что говорится в лоб. Для простоты рассмотрим случай работы только с файлами *.vbs. С этой цель в папке htm создадим файл html станицы, который имеет следующий вид:

Файл client_file.htm

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=utf-8">
<title>Send File To WebSocket</title>
<script type="text/javascript">
  function readFile() {
    var selectFile = document.getElementById('f').files[0];
    var reader = new FileReader();
    reader.onload = function (e) {
      var FileContent = e.target.result;
      parseContent(FileContent);
    };
    reader.readAsText(selectFile);
  }
  function parseContent(content) {
     document.all("txt").value = content;
  }
</script>
</head>
```

```
<body>
  <input id="f" type="file" onchange="readFile()"><br />
  <textarea id="txt" cols="80" rows="20"></textarea><br />
  <input id="btn" type="button" value="SEND TO SERVER : "/>
  <pre id="log"></pre>

<script type="text/javascript">
  btn.addEventListener('click', function(e) {
     var ws = new WebSocket('ws://192.168.200.203:8081');
     ws.onopen = function() {
       log('CONNECTED');
       ws.send(txt.value.replace(/\n/g, ':')+'\n');                    };
         ws.onmessage = function(e) {
       log('MESSAGE: ' + e.data);
     };
         ws.onclose = function(e) {
       log('DISCONNECTED: code = ' + e.code);
         };
  });
   function log(msg) {
         document.getElementById('log').innerHTML += '<br>'+msg;     }
</script>
</body>
</html>
```

Запустив Web сервис, используя команду RunServerVB и подключившись к нему из браузера, появляется возможность выбрать одну из html страниц, хранимых на сервере в папке htm. Выбрав файл client_file.htm, в браузер будет загружена страница, в которой кнопка выбора позволяет в текстовую область страницы загрузить нужный скриптовый файл. Этот код, если нужно, может быть отредактирован и отправлен для выполнения на сервер, чтобы обратно получить результат его работы (рис. 5).

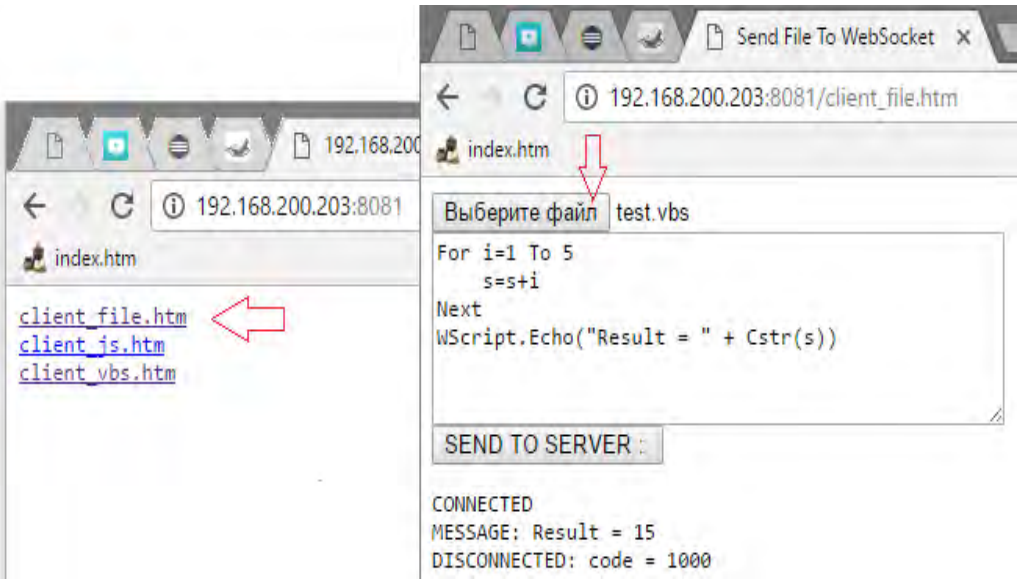


Рис. 5. Ввод и результат выполнение JS скрипта в среде Google Chrome

Рассмотренный выше подход позволяет использовать WebSocketD как инструмент удаленного администрирования. Чтобы показать возможность использования различных компонентов, используемых при удаленном администрировании, создадим файл test_admin.vbs.

Файл test_admin.vbs

```
Set oWshNet = WScript.CreateObject("WScript.Network")
WScript.Echo("CompName: " & oWshNet.ComputerName)

Set oShell = CreateObject("Wscript.Shell")
WScript.Echo("CurDir  : " & oShell.CurrentDirectory & "\")

Set oWMI = GetObject("winmgmts:\\127.0.0.1\root\cimv2")
sQry = "Select * From Win32_DesktopMonitor"
Set colItems = oWMI.ExecQuery(sQry)
For Each oItem In colItems
   Wscript.Echo "Display : " & oItem.ScreenWidth & " x " & oI-
tem.ScreenHeight
Next

sQry = "Select * from Win32_OperatingSystem"
Set colItems = oWMI.ExecQuery(sQry)
For Each oItem In colItems
   Wscript.Echo "OS ver. : " & oItem.Version
Next
```

Сначала проверим работу этого скрипта, при условии, что сервер запущен на локальном компьютере с операционной системой Windows 10.

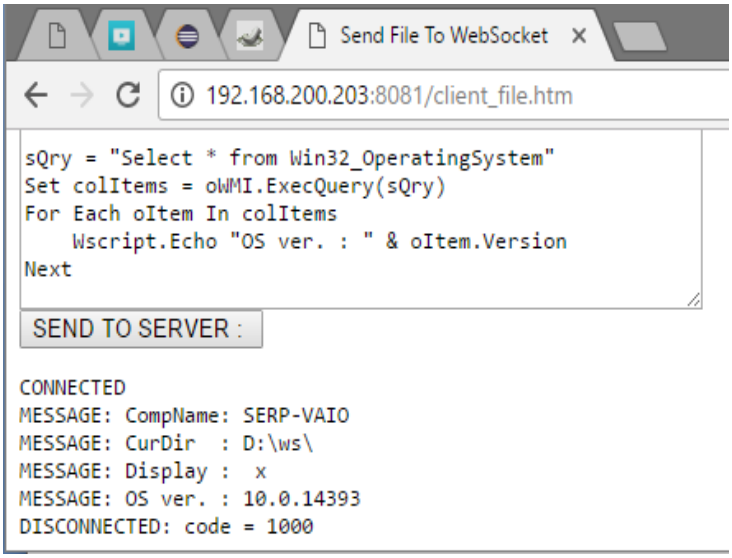


Рис. 6. Ввод и результат выполнение JS скрипта в среде Google Chrome

А потом на удаленном ПК, в качестве которого будет использована виртуальная машина с ОС Windows XP и 32-разрядной версией WebSocketD совместно с WHS-сервером. Для этого эксперимента следует использовать файлы, аналогичные Server1.vbs, RunServerVB и client_file.htm, но несколько видоизмененные и со своими путями доступа к файлам, которые могут отличаться от тех, что были на локальном компьютере.

Выполняя команду ipconfig, можно узнать IP адрес виртуальной машины, который имеет значение 192.168.200.202. К Web сервису этого адреса надо подключить клиента, загрузить файл test_admin.vbs и нажать SEND TO IP. Скрипт будет передан на виртуальную машину XP-VM, там выполнен, а результаты будут возвращены клиенту и там же отображены (рис. 7).

Из анализа результатов можно узнать имя компьютера, текущий каталог, из которого запущена утилита WebSocketD, версию ОС, установленную на XP-VM, а также разрешение ее экрана.

Если теперь уменьшить размер окна виртуальной машины, а в браузере основного компьютера повторно нажать кнопку SEND TO IP, то повторный результат выполнения скрипта даст несколько иной результат:

```
CONNECTED
MESSAGE: CompName: XP-VM
MESSAGE: CurDir  : F:\Transas\ws\Sample_4\
MESSAGE: Display : 671 x 517
MESSAGE: OS ver. : 5.1.2600
DISCONNECTED: code = 1000
```

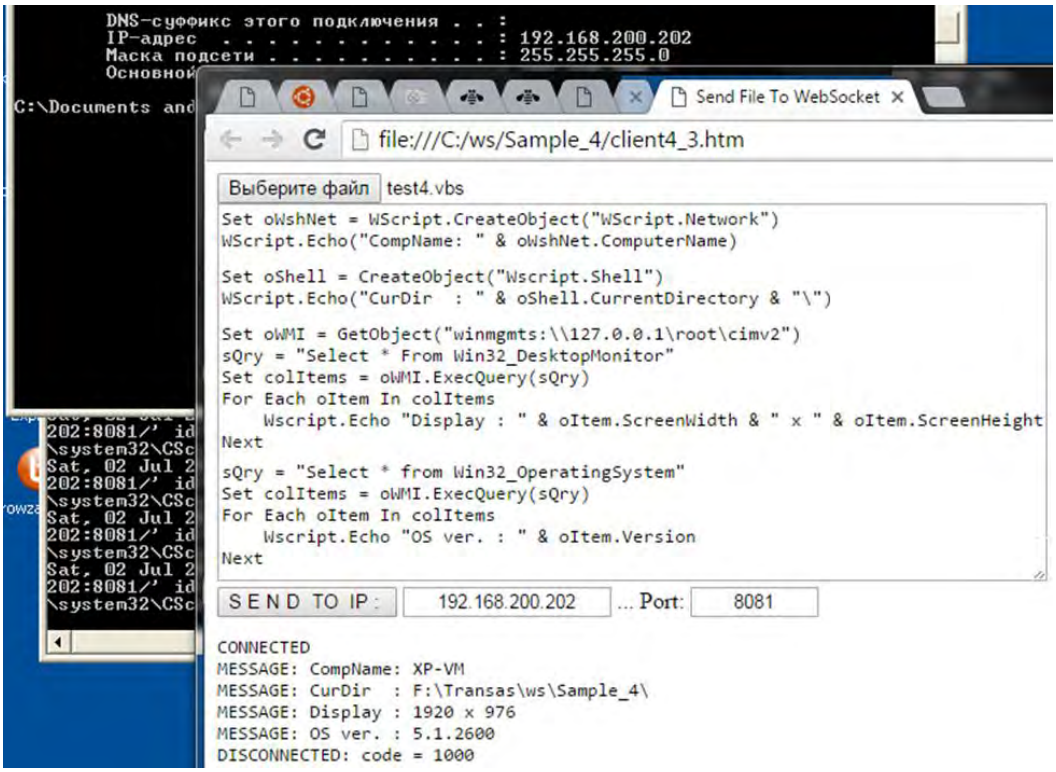


Рис. 7. Выполнение скрипта на виртуальной машине с ОС Windows XP

Подводя некоторый итог, следует отметить возможности использования утилиты WebSocketD по удаленному администрированию и выполнению скриптовых кодов. Однако, надо отметить и некоторый недостаток, который связан с тем, что для выполнения кодов на Jscript и VBScript требовалось разрабатывать и использовать разные как серверные, так и клиентские коды.

Общий Web сервис для JScript и VBScript кодов

В данном разделе будет предложен один из возможных подходов для реализации единых как серверных, так и клиентских кодов, позволяющих удаленно выполнять скрипты на любом из поддерживаемых WSH языков.

Наличие в браузере клиента возможности на html странице выбирать и загружать в ее текстовую область файл по его имени, позволяет определить тип этого файла по его расширению.

Зная расширение файла, можно изменять выходной поток с клиента на сервер в зависимости от типа загруженного файла. Попробуем учесть эту возможность в новой интерфейсной html странице, которая получена из файл client_file.htm путем минимальных изменений и добавлений. Эти коррекции выделены на приведенном ниже листинге более темным цветом:

Файл client_wsf.htm

```
. . .
 <script type="text/javascript">
   var ext;
         function readFile() {
```

```
      var selectedFile = . . .;
                FileName = selectedFile.name;
                ext = FileName.split('.').pop().toLowerCase();
      var btnSend = document.getElementById("btn")
      btnSend.hidden = !(ext=='vbs' || ext=='js')
      var reader = new FileReader();
. . .
<body>
   Загрузка JS- и VBS- файлов в кодировке UTF-8
         для выполнение их сервере <br /><br />
. . .
   var ws = new WebSocket('ws://192.168.200.203:8083');
         ws.onopen = function() {
           log('CONNECTED');
     //ws.send(txt.value.replace(/\n/g, ':')+'\n');
           if (ext == 'vbs'){
                ws.send('vb'+txt.value.replace(/\n/g,':')+'\n');
           } else {
       ws.send('js'+txt.value.replace(/\n/g,'')+'\n');
           }
         };
```

Переменная FileName содержит имя выбранного файла, а переменная ext его расширение. Если расширение файла не соответствует *.vbs или *.js, то кнопка отправки файла становится невидимой, что исключает возможность передачи на сервер файлов, которые не могут быть выполнены.

Предполагая, что новый Web сервис может работать совместно с ранее разработанными сервисами, зарезервируем за ним порт 8083. Именно на этот порт будет направляться выходной поток клиента, структура которого будет изменяться в зависимости от типа файла, что предусматривает посылку в начале каждого потока двух дополнительных символов, характеризующих язык программного кода, передаваемого на сервер.

Разобравшись с интерфейсной частью клиента и его выходным потоком можно перейти к серверной части сервиса, который должен принять поток от клиента, выделить его первые два символа, чтобы узнать язык программного кода, поступившего от клиента, и выполнить его.

Следует отметить, что сервер сценариев WSH — это компонент Microsoft Windows, который предназначен для запуска сценариев на ряде скриптовых языков. Причем не только на JScript или VBScript, но и на ряде других дополнительно устанавливаемых языках. Таких, например, как Perl.

Особенность WSH в том, что он дает возможность комбинировать все установленные в системе скриптовые языки в одном файле с расширением .wsf. Именно такой файл предлагается использовать как серверный код Web сервиса для удаленного выполнения программ на любом из двух языков.

Файл Server3.wsf (Файл в кодировка UTF-8)

```
<?xml version="1.0" encoding="UTF-8"?>
<job id="MyTestJob">

   <script language="VBScript">
   <![CDATA[
            Function ExecuteVBS(cod_vbs)
                  ExecuteVBS = Execute(cod_vbs)
     End Function
   ]]>
   </script>

   <script language="JScript">
   <![CDATA[
            var DataIn = WScript.StdIn.ReadLine();
            var id = DataIn.substring(0,2);
            var cod = DataIn.substring(2);
               if (id == 'vb') {
                     ExecuteVBS(cod);
               } else if (id == 'js') {
                     eval(cod);
               } else {
                     WScript.Echo('Error in wsf-file !' + '\n');
               }
   ]]>
   </script>
</job>
```

Основную работу по приему входного потока выполняет модуль Jscript, который разделяет его на две части: первые два символа сохраняются в id, а оставшаяся часть принятого потока записывается в переменную cod. В том случае, когда идентификатор потока будет равен 'js' поступивший от клиента программный код будет выполнен внутри модуля Jscript.

Если идентификатор потока будет иметь значение 'vb', то в этом случае внутри модуля Jscript будет вызываться функция ExecuteVBS(cod), которая определена в модуле VBScript.

Запуск Web сервиса с таким серверным wsf-кодом ничем не отличается от рассмотренных ранее подходов. Командный файл для этого случая будет иметь вид:

Файл RunServerWSF.cmd

```
websocketd --port=8083 --staticdir=.\htm CScript /nologo Server3.wsf
```

После того как этот файл будет запущен на сервере имеется возможность на любом удаленном компьютере открыть браузер, подключиться к Web сервису, выбрать интерфейсный файл client_wsf.htm, а после этого загрузить любой из файлов *.vbs или *.js, если нужно отредактировать, и отправить на сервер для выполнения (рис. 8).

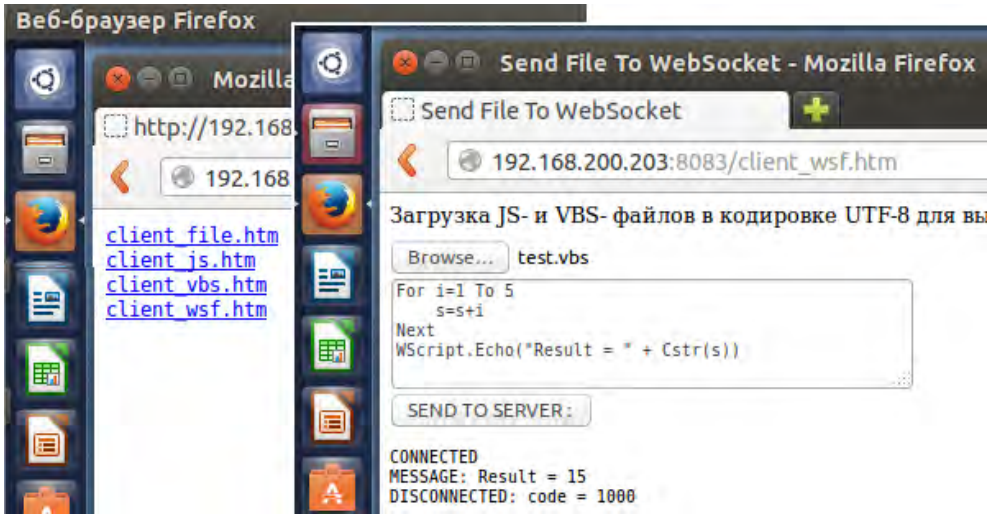


Рис. 8. Выполнение VBS скрипта на компьютере с ОС Ubuntu

С использование WebSocketD можно на одном сервере создать несколько Web сервисов, обеспечив множественные подключения к каждому из них, а сама серверная обработка, в отличии от рассмотренных примеров, может использовать достаточно мощные программные продукты.

Библиографический список

1. С.П.Хабаров. Организация гетерогенных ЛВС с терминальным доступом между ее узлами. //Известия СПбЛТА-СПб .,2016. № 216, с. 267-280
2. С.П.Хабаров. Терминальный доступ между UBUNTU и WINDOWS компьютерами. //Информационные системы и технологии: теория и практика: сб. научн. тр.-СПб. .:СПбГЛТУ ,2016. № 8, с. 30-43.

С.П. Хабаров, кандидат технических наук, доцент

ВЗАИМОДЕЙСТВИЯ УЗЛОВ СЕТИ ПО ПРОТОКОЛУ WEBSOCKET

Развитие протокола WebSocket явилось ответом на необходимость в более эффективных решениях построения Web-приложений, работающих по технологии клиент-сервер в реальном времени. До появления WebSocket использовались такие методы, как периодический опрос сервера (Polling) или очередь ожидающих запросов (Long Polling).

Оба метода имеют свои преимущества, но имеют общие недостатки. Они используют протокол HTTP, чтобы отправлять сообщения на сервер. При этом каждый пакет оборачивается массой заголовков, которые описывают, где пакет движется, откуда он пришел, информацию о браузере и прочее. Все это добавляет много накладных расходов при работе в реальном времени. Ни один из этих методов не является "двунаправленным полным дуплексом", когда клиент и сервер могут посылать и получать информацию друг от друга одновременно.

По этим причинам они не очень хороши для быстрой, масштабируемой связи в реальном времени в Интернете. От этих недостатков более свободен протокол WebSocket и построенные на его основе Web сервисы. Достоинство протокола и в том, что он имеет достаточно простой в использовании API [1], который работает в современных браузерах разных ОС [2].

В данной статье основное внимание будет уделено собственно протоколу WebSocket, его спецификации и взаимодействию узлов сети, использующих этот протокол. С этой целью рассматривается пример подключения клиента с помощью браузера к WebSocket-серверу и захват утилитой Wireshark всех TCP фреймов, которыми клиент и сервер будут обмениваться между собой.

Процесс взаимодействия по протоколу WebSocket

Протокол WebSocket позволяет решать широкий круг задач, снимая ряд ограничений на обмен данными между браузером и сервером, обеспечивая достаточную безопасность и существенное сокращение сетевого трафика.

Для двухсторонней коммуникации он требует, чтобы и клиентское, и серверное приложения поддерживали все детали этого протокола. То есть нужна, например, WebSocket-совместимая html страница, которая вызывает WebSocket-совместимую конечную точку.

При взаимодействии узлов сети по протоколу WebSocket выделяют три основных процесса:

Взаимодействие начинается с процесса установления связи (handshake), в ходе которого браузер и сервер подтверждают свое намерение взаимодействовать по постоянному соединению.

- Далее по TCP в обе стороны посылается серия пакетов сообщений, причем посылки могут выполняться асинхронно, и в разных объемах.
- При закрытии соединения обе конечные точки обмениваются кадром закрытия (close frame) для корректного завершения связи.

Рассмотрим процесс взаимодействие узлов сети по протоколу WebSocket на примере подключения простой html страницы (рис. 1) из браузера Google Chrome к стандартному WebSocket эхо-серверу (ws:\\echo.websocket.org).

1_test_www.htm

```
<!DOCTYPE html>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<textarea id="txt" cols="60" rows="10"></textarea><br/>
<input id="btn" type="button" value="R U N"/>
<pre id="log"></pre>
<script>
   btn.addEventListener('click', function(e) {
       var ws = new WebSocket('ws://echo.websocket.org/');
       ws.onopen = function() {
           log('CONNECTED');
           ws.send(txt.value);
       };
       ws.onmessage = function(e) {
           log('MESSAGE: ' + e.data);
       };
       ws.onclose = function(e) {
           log('DISCONNECTED: code = ' + e.code);
       };
   });
   function log(msg) {
       document.getElementById('log').innerText += msg + '\n';
   }
</script>
```

Рис. 1. Код html станицы для подключения к ws:\\echo.websocket.org

После загрузки этой html страницы в браузер, ввода произвольного текста в ее текстовую область, нажатия кнопки RUN и получения сообщения от сервера, страница будет иметь вид, аналогичный приведенному на рис. 2.

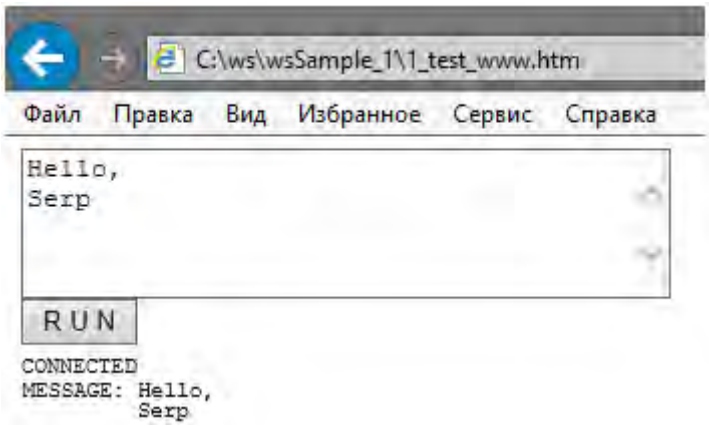


Рис. 2. Вид станицы после подключения к ws:\\echo.websocket.org

Если при этом, используя утилиту Wireshark, выполнить захват сетевого трафик между браузером и сервером, а затем отфильтровать все захваченные пакеты для протоколов WebSocket и HTTP, то можно явно различить все три процесса взаимодействия между браузером и сервером при WebSocket-соединении (рис. 3).

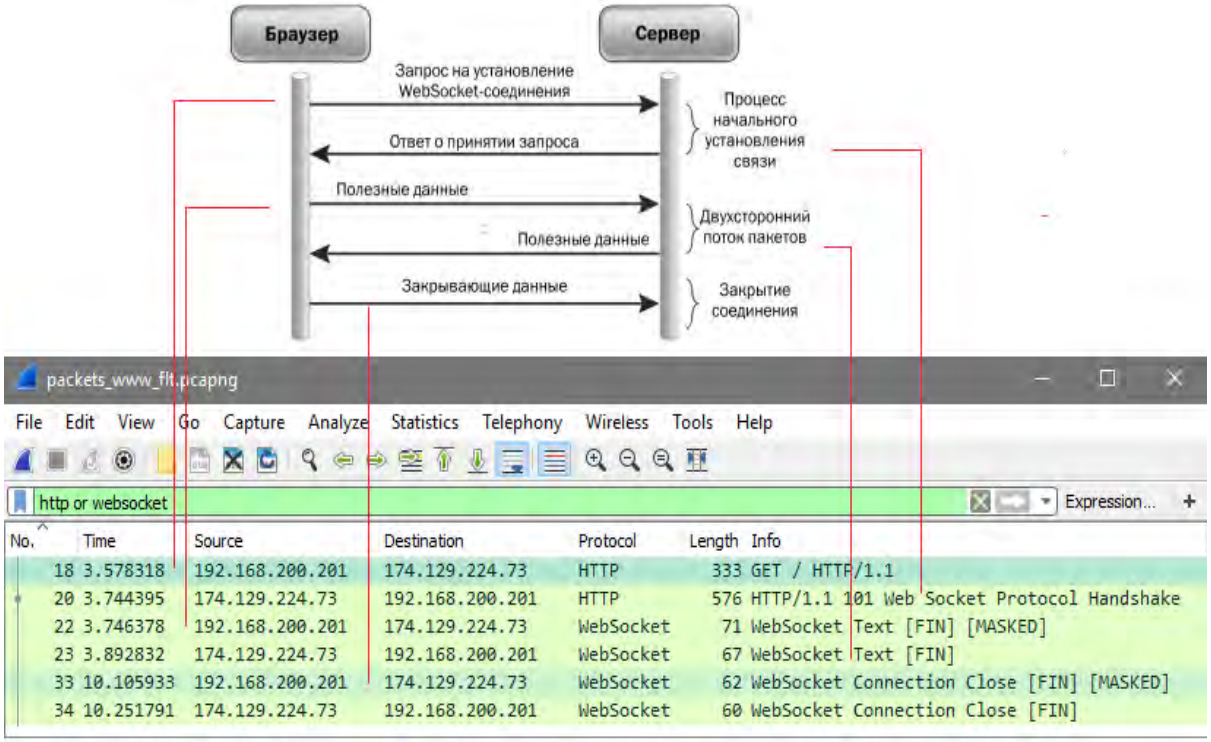


Рис. 3. Взаимодействие браузера и сервера при WebSocket-соединении

На основе этого примера рассмотрим взаимодействие браузера и сервера при WebSocket-соединении, раскрывая и тщательно исследуя каждый из шести фреймов, которыми обменялись между собой браузер и сервер.

Протокол WebSocket был разработан так, чтобы успешно работать в уже существующей веб инфраструктуре. Поэтому его спецификация определяет, что соединение WebSocket начинает свою жизнь как HTTP соединение, гарантируя при этом полную обратную совместимость.

Установление WebSocket-соединения

Протокол WebSocket работает над протоколом HTTP, поэтому браузер при подключении к серверу отправляет ему специальные HTTP заголовки, с запросом о поддержке сервером протокола WebSocket. Если сервер ответит положительно, то HTTP соединение прекращается и дальнейшее общение идёт по протоколу WebSocket, который с HTTP уже не имеет ничего общего.

Рассмотрим процесс установления соединения более подробно. В самом начале этого процесса клиент формирует и посылает Web серверу обычный HTTP-запрос, который представляет собой команду HTTP GET.

Этот запрос сконфигурирован как запрос переключения (upgrade request), основное отличие которого от обычного HTTP запроса состоит в наличии в его составе дополнительных заголовков.

В рассматриваемом примере этому HTTP-запросу соответствует фрейм №18, раскрыв который в среднем окне Wireshark можно увидеть полное его содержимое (рис. 4).

Наличие в запросе клиента заголовка Upgrade указывает на то, что клиент просит сервер переключиться на другой протокол. Кроме этого в запросе клиента используются следующие заголовки:

- GET, Host – это стандартные HTTP заголовки из URL запроса.
- Connection, Upgrade – эти заголовки указывают на то, что браузер хочет перейти на другой протокол.
- Origin – здесь указан протокол, домен и порт, откуда отправлен запрос.
- Sec-WebSocket-Key – в этот заголовок помещается случайный ключ, который генерируется браузером: 16 байт в кодировке Base64.

Sec-WebSocket-Version – это версия протокола, текущая версия: 13.

Все заголовки, кроме GET и Host, браузер сгенерирует сам. Возможности что-либо изменить средствами скрипта html страницы не предусмотрено.

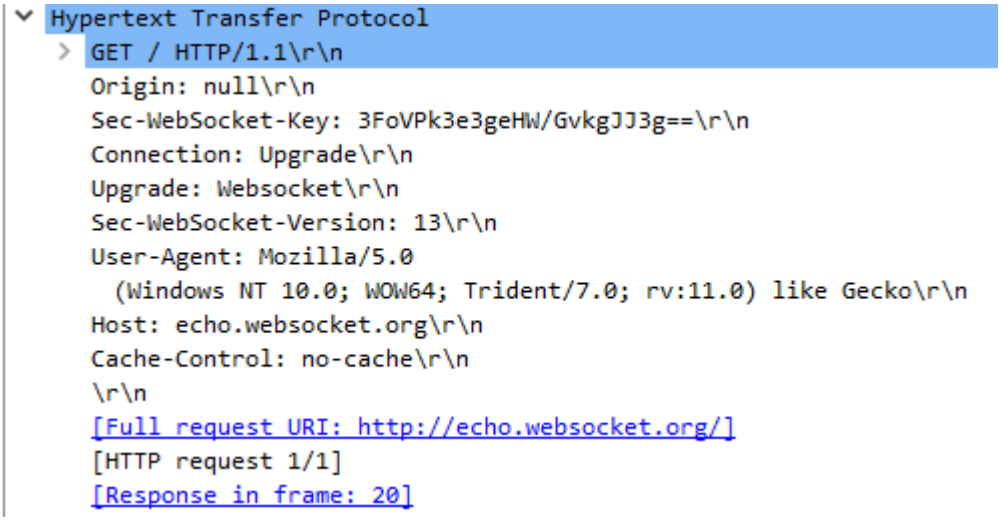


Рис. 4. HTTP-запрос клиента (фрейм №18)

При анализе запроса клиента, сервер принимает решение о допустимости WebSocket с узла, адрес которого указан в заголовке Origin. Поэтому при отправке на сервер клиентского запроса, в нем всегда должен присутствовать этот заголовок.

В общем случае клиент может посылать другие заголовки, кроме тех, что были перечислены выше. Возможны такие дополнительные заголовки, как Sec-WebSocket-Extensions и Sec-WebSocket-Protocol. Они будут описывать расширения и подпротоколы, которые может поддерживать данный клиент.

Что касается рассматриваемого примера, то здесь следует отметить, что запрос к серверу браузер сформирует самостоятельно, выполняя скрипт html страницы, когда будет создаваться новый объект:

```
new WebSocket("ws://echo.websocket.org");
```

Если сервер, приняв запрос клиента, решает разрешить ему подключение по протоколу WebSocket, то он вернет клиенту HTTP ответ, который будет аналогичен тому, что в рассматриваемом примере получен с узла, IP адрес которого 174.129.224.73 и содержится во фрейме №20 (рис. 3). Раскрыв его содержимое имеет вид, который представлен на рис. 5.

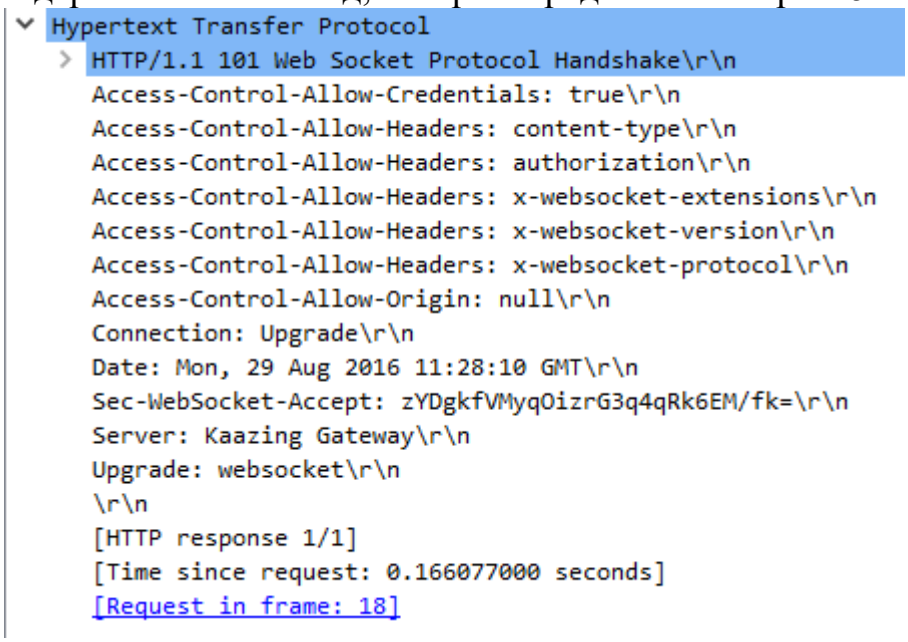


Рис. 5. HTTP-ответ сервера (фрейм №20)

Первая строка ответа сервера сообщает, что “рукопожатие” успешно завершено. Код статуса 101 в протоколе HTTP/1.1 говорит о том, что сервер предлагает перейти на более подходящий для указанного ресурса протокол, список которых он указывает в поле заголовка Upgrade.

Строка с заголовком Sec-WebSocket-Accept объявляет, что сервер готов к подключениям, и сообщает специальным образом вычисленный хэш от ключа, переданного в запросе клиента в заголовке Sec-WebSocket-Key. Это значение не является принципиальным при работе стандартного браузера со стандартным WebSocket сервером. Однако, при разработке своего сервера всегда надо вычислять этот хэш, используя алгоритм криптографического хеширования SHA-1.

Из рис. 5 видно, какое значение сервер отправляет клиенту в заголовке sec-WebSocket-Accept в ответ на полученный от него ключ Sec-WebSocket-Key. Клиент в виде браузера использует это значение для проверки того, что ответ предназначается именно ему.

После “рукопожатия” соединение останется открытым, и узлы могут друг с другом общаться по двунаправленному полному дуплексу, а данные будут передаваться по протоколу, структура которого («фреймы») изложена далее.

Формат данных протокола WebSocket

Стандарт этого протокола утверждён международной организацией IETF (Internet Engineering Task Force – Инженерный совет Интернета), а полное его описание содержится в RFC 6455

(https://tools.ietf.org/html/rfc6455). Здесь рассмотрим только краткое описание его самых важных частей.

В протоколе WebSocket предусмотрены несколько видов фреймов, среди которых выделяют два больших класса. Это фреймы с данными («data frames») и управляющие фреймы («control frames»), которые используют для проверки связи (PING, PONG) и закрытия соединения. Общий вид формата фрейма протокола WebSocket приведен на рис. 6.

В заголовке фрейма сначала идут биты FIN, RSV1, RSV2, RSV3, следом за ними «Опкод» (4 бита), «МАСКА» (1 бит) и «Длина сообщения» (7 бит). Если значение в поле «Длина сообщения» будет равно 126 или 127, то в посылке появится поле «Расширенная длина тела», размером в 16 или 64 бита. Заканчивают посылку «Ключ маски» (4 байта) и само сообщение.

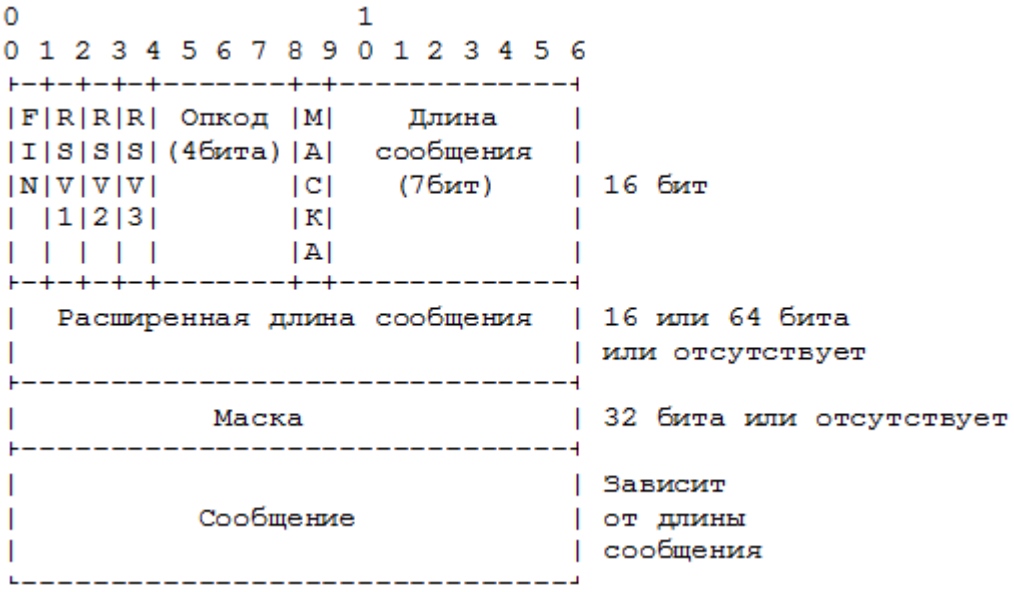


Рис. 6. Формат фрейма протокола WebSocket

Рассмотрим подробнее описание отдельных частей фрейма, которые определяют, как именно передаются сообщения:

- «FIN» – Если сообщение длинное, то оно фрагментируется. У всех фреймов, кроме последнего, FIN=0. У последнего FIN=1. Если сообщение состоит всего из одного фрейма, то в нём FIN равен 1.«RSV1, RSV2, RSV3» – Они предназначены для расширений протокола и обычно их значения равны 0.
- «Опкод» – Его значение определяет тип данных, находящихся во фрейме:

  0x1 – обозначает текстовый фрейм.

  0x2 – обозначает двоичный фрейм.

  0x3 - 0x7 – зарезервированы для будущих фреймов с данными.

  0x8 – обозначает закрытие соединения этим фреймом.

  0x9 – обозначает PING.

  0xA – обозначает PONG.

- «Маска» – Если этот бит равен 1, то данные фрейма маскированы.
- «Длина сообщения» – Значения от 0 до 125 – интерпретируется, как длина сообщения. Если 126, то следующие 2 байта интерпретируются как 16-битное беззнаковое целое число, содержащее длину сообщения. Если значение поля равно 127, то следующие 8 байт интерпретируются как 64-битное беззнаковое целое, содержащее длину сообщения.

  Такая схема позволяет минимизировать накладные расходы. Если длина сообщений до125 байт, то для хранения его длины нужно всего 7 бит, для сообщений от 126 до 65536 байт – 7 битов + 2 байта, а для совсем больших – 7 битов и 8 байт. Этого хватит для хранения длины сообщения размером в гигабайт и более.
- «Ключ маски» – Если бит Маска = 0, то этого поля не будет, если он равен 1, то эти 4 байта содержат маску, которая налагается на тело.
- «Данные фрейма (сообщение) » – Их длина должна быть равна той, что указана в заголовке.

После того, как связь установлена клиент и сервер могут обмениваться фреймами протокола WebSocket, в котором “полезные” данные начинаются с операционного кода («Опкод»). Значение этого поля указывает тип операции, которую надо выполнить в процессе обработки конкретного фрейма.

Надо отметить, что WebSocket-сообщения передаются асинхронно, поэтому на отправленный запрос не обязательно будет получен мгновенный ответ. При использовании протокола WebSocket лучше мыслить в категориях универсальных сообщений, передаваемых от клиента серверу и обратно, и забыть о традиционном шаблоне запроса/ответа в HTTP.

Процесс обмена WebSocket сообщениями

Разобравшись со структурой фрейма протокола WebSocket, вернемся к рассматриваемому примеру, где уже установлено соединение, и браузер начинает передавать данные на сервер. Этому процессу в списке пакетов, которые были захвачены Wireshark (рис. 3), соответствует фрейм №22.

Именно он обеспечивает передачу на сервер двух текстовых строк, которые были введены на html странице до нажатия RUN. Если в Wireshark раскрыть этот фрейм, то можно обнаружить структуру, которая полностью аналогична описанной выше, и имеет вид, приведенный на рис. 7.

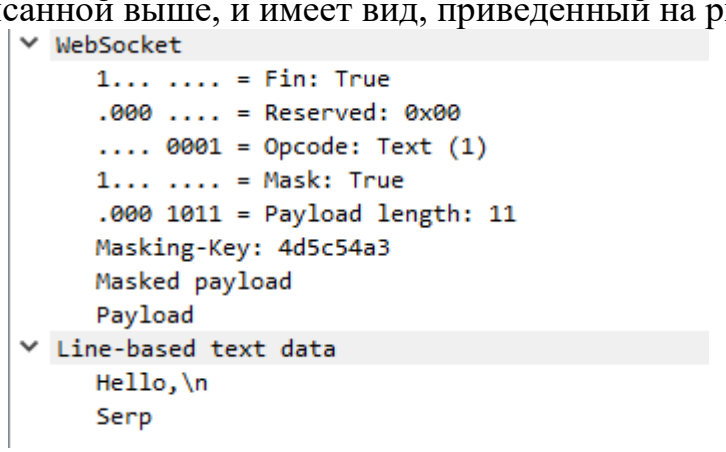


Рис. 7. Фрейм передачи данных от клиента на сервера (фрейм №22)

Из рис. 7 видно, что осуществляется передача одного единственного (Fin=1) текстового фрейма (Opcode=0x1) протокола WebSocket с длиной тела сообщения всего в 11 байт. Причем содержание текстовых данные в точности соответствуют тем, что вводились в окне html-страницы. Эти данные явно не присутствуют в структуре фрейма, но имеются в расшифровке Line-based text data утилиты Wireshark.

Если обратиться непосредственно к структуре фрейма, то в поле Masked payload вместо последовательности кодов 48-65-6c-6c-6f-2c-0a-53-65-72-70, соответствующих исходному тексту, будет совсем иная последовательность из 11 байт – 05-39-38-cf-22-70-5e-f0-28-2e-24 (рис. 8).

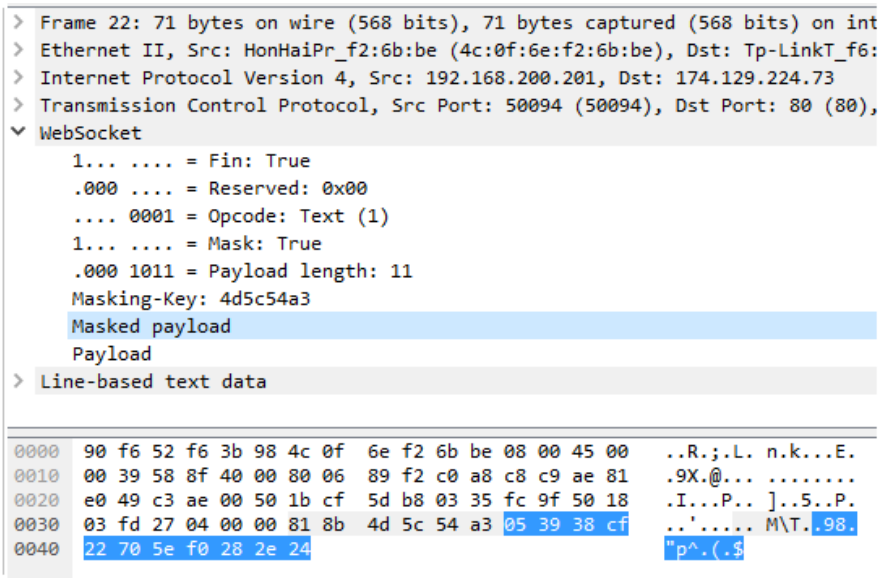


Рис. 8. Фрагмент masked payload фрейма передачи данных на сервер

Это отличие станет понятно, если обратить внимание на то, что браузер в структуре фрейма WebSocket установил флаг маски (Mask=1) и задал ключ маски («Masking-Key»=4d5c54a3). Это значение WebSocket использует для маскирования сообщения, что должно защитить от атак типа «отравленный кэш». При этом наблюдаются следующие различия:

- Фрейм, поступающий от браузера, как правило, замаскирован.
- Фрейм, поступающий от сервера, не должен быть замаскирован.

При значениях Mask=1 над каждым байтом сообщения выполняется побитовое «исключающее или» с соответствующим байтом маски (рис. 9).

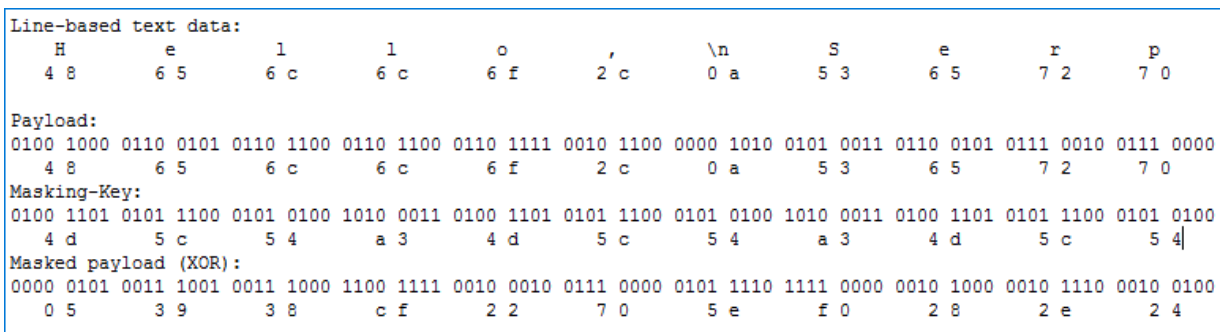


Рис. 9. Пример получения замаскированного сообщения

Именно это и наблюдается при формировании фрейма передачи данных на сервер. В отличие от этого, как видно из рис. 10, фрейм, поступивший с сервера, не имеет флага маски (Mask=0), а поле Masking-Key в его структуре вообще отсутствует.

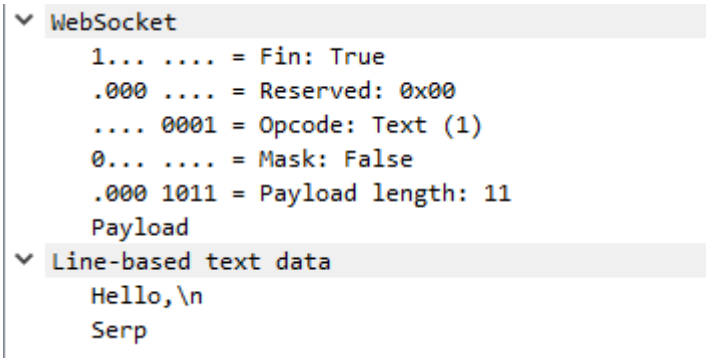


Рис. 10. Фрейм, поступивший клиенту с сервера (фрейм №23)

Процесс асинхронной передачи и приема сообщений может выполняться достаточно долго, пока одна из сторон не закроет это соединение. Так как в примере использована простейшая html страница, то средств нормального закрытия соединения в ней не предусмотрено.

Единственный вариант прервать соединение с сервером – это просто закрыть html страницу в браузере. Несмотря на аварийное прерывание соединения по протоколу WebSocket клиент и сервер успевают обменяться сообщениями, в которых фрагменты фрейма «Опкод» равны 0x8 (рис. 11).

Следует отметить, что и в этом случае фрейм клиента замаскирован, а передаваемое им сообщение соответствует коду завершения, который в этой ситуации принимает значение 1001 (удалённая сторона «исчезла»). Такое значение код завершения принимает в том случае, когда процесс сервера «убит» или браузер перешёл на другую страницу. При нормальном закрытии соединения код завершения равен 1000.

.

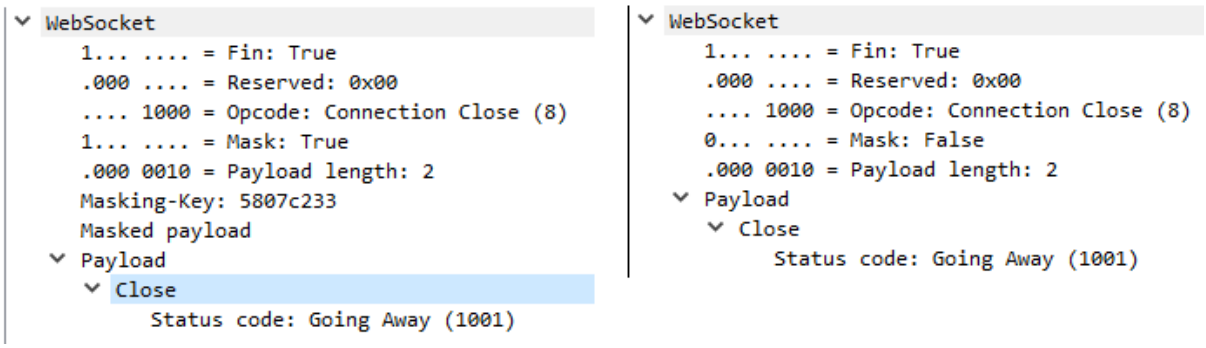


Рис. 11. Фреймы завершения соединения (фреймы №33 и №34)

Асинхронность обмена сообщениями в WebSocket

Из приведенного примера становится ясно, что единожды установленное соединение дает возможность клиенту и серверу постоянно обмениваться сообщениями. Причем, как указано в спецификации протокола, еще и в асинхронном режиме.

Чтобы проиллюстрировать этот процесс рассмотрим небольшой пример, в котором клиент через некоторую паузу посылает на сервер три команды. После поступления от сервера подтверждения о получения команды каждая из них обрабатывается путем нескольких взаимодействий между клиентом и сервером. Для реализации этой задачи выполним простейшую модификацию html страницы исходного примера:

```
. . .
  ws.onopen = function() {
        ws.send('cmd1-');
        setTimeout(function() { ws.send('cmd2-') }, 100);
 setTimeout(function() { ws.send('cmd3-') }, 200);
  };
  ws.onmessage = function(e) {
        log('MESSAGE: ' + e.data);
        if (e.data.length == 9) { ws.close(); }
        else { ws.send(e.data + 'a'); }
  };
. . .
```

После загрузки этой html страницы в браузер, и нажатия кнопки RUN, будет установлено соединение с WebSocket сервером и выполнится обмен сообщениями между клиентом и сервером, который носит явно асинхронный характер (рис. 12).

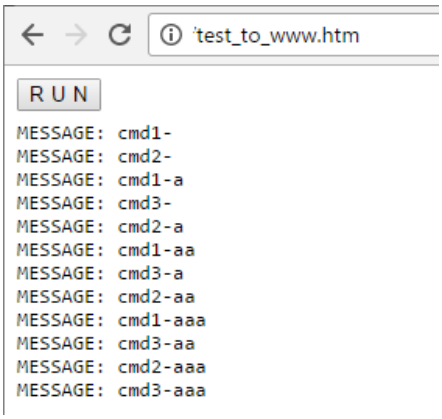


Рис. 12. Результат работы примера Web приложения

Данный пример показывает возможность использования протокола WebSocket для реализации интерактивных Web-приложений, входящих в состав различных автоматизированных систем управления [3, 4]. При этом под интерактивностью будем понимать:

- Минимальную латентность, при которой человек должен получать реакцию на свое действие как можно быстрее, обеспечивая принцип «нажми на кнопку — получишь результат».
- Асинхронность, при которой текущая деятельность клиента не должна мешать реакции на вновь поступающие сообщения. Если это условие не выполняется, то интерактивность теряется

Чтобы интерактивное Web приложение работало так же быстро, как и десктопное приложение, надо предусмотреть такой обмен между сервером и клиентом, чтобы он осуществлялся посредством маленьких, максимально частых сообщений. При таком подходе будет минимизирована латентность, и может быть обеспечена асинхронность.

Библиографический список

1. Хабаров С.П. Использование утилиты websocketd для удаленного выполнения программ. //Сборник научных трудов "Информационные системы и технологии: теория и практика" – СПб.:СПбГЛТУ, 2017. № 9,
2. Хабаров С.П.. Организация гетерогенных ЛВС с терминальным доступом между ее узлами. //Известия Санкт-Петербургской лесотехнической академии – СПб.:СПбГЛТУ, 2016, Вып.216., с. 267-280.
3. Заяц А. М., Логачев А. А. Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей. //Известия Санкт-Петербургской лесотехнической академии – СПб.:СПбГЛТУ, 2016, Вып. 216, с. 241-255.
4. Амбросовский В.М., Баглюк Ю.В., Слипченко А.С., Хабаров С.П. Интегрированные системы управления техническими средствами корабля. Морской вестник. 2014, № 2 (50), с. 63-65.

С.П. Хабаров, кандидат технических наук, доцент
К.С. Голубев, студент 2 курса магистратуры

КЛИЕНТ-СЕРВЕРНАЯ ЭКСПЕРТНАЯ СИСТЕМА НА ОСНОВЕ ТЕХНОЛОГИИ WEBSOCKET

Благодаря широкому развитию Интернет в XXI веке стало возможным удаленно обрабатывать данные, которые хранятся на серверах, обеспечивая доступ к ним помощью стандартных браузеров, входящих в состав широкого класса операционных систем современных вычислительных устройств [1].

В данной статье рассмотрен один из возможных подходов к организации клиент-серверной экспертной системы (ЭС) на основе технологии websocket . Он иллюстрируется примером разработки ЭС «Определитель хвойных пород по виду шишек», которая позволяет пользователю, заменяя эксперта, получать вывод о том, какая хвойная порода находится перед

ним на основе внешнего вида ее шишек. Принцип работы системы основан на удаленном соединении с сервером и прохождении консультации по каналу websocket.

Начало разработки ЭС относят к 1970-х годам, а в 1980-х они получили коммерческое подкрепление. Предтечи экспертных систем были предложены еще в 1832 году С. Н. Корсаковым, создавшим механические устройства, так называемые «интеллектуальные машины». Они позволяли находить решения по заданным условиям, например, определять наиболее подходящие лекарства по наблюдаемым у пациента симптомам заболевания.

В настоящее время экспертные системы рассматриваются совместно с базами знаний как модели поведения экспертов в определенной области знаний с использованием процедур логического вывода и принятия решений, а базы знаний — как совокупность фактов и правил логического вывода в выбранной предметной области деятельности [2].

WebSocket — это следующее поколение технологии двунаправленнойсвязи между браузером и сервером, когда между пользователем и сервером устанавливается постоянное подключение, в котором обе стороны могут обмениваться данными в реальном времени [3,4]. Этот протокол работает над HTTP, что позволяет пересылать любые данные, на любой домен, безопасно и почти без лишнего сетевого трафика.

На основе этого протокола появляется возможность строить архитектуру клиент-серверных ЭС по принципу тонкого клиента, при котором на клиенте нет никакой обработки данных, а все выполняется на сервере. В качестве простейшего HTTP и WS сервера можно использовать утилиту WebSocketD, которая будет:

- принимать внешнее http подключение;
- пересылать клиенту стартовую html-страницу, которая содержит js-код, обеспечивающий переход на ws-соединение;
- поддерживать ws-соединение с клиентом, обеспечивая переключение входных и выходных внешних потоков на серверное ядро ЭС.

В качестве программного обеспечения серверного ядра ЭС предлагается использовать консольную реализацию оболочки продукционных ЭС CLIPS, а также ряд баз знаний (БЗ), реализованных в этой среде (файлы *.clp). Все они представляет собой набор фактов и правил, на основе которых ЭС может формулировать заключение.

Если в составе системы предполагается использование и нечетких БЗ, то в этом случае в серверное ядро ЭС может быть дополнительно подключена еще и такая оболочка ЭС, как FuzzyCLIPS. Для реализации предложенного подхода к созданию клиент-серверной ЭС будет использоваться архитектура, аналогичная той, что приведена на рис.1.

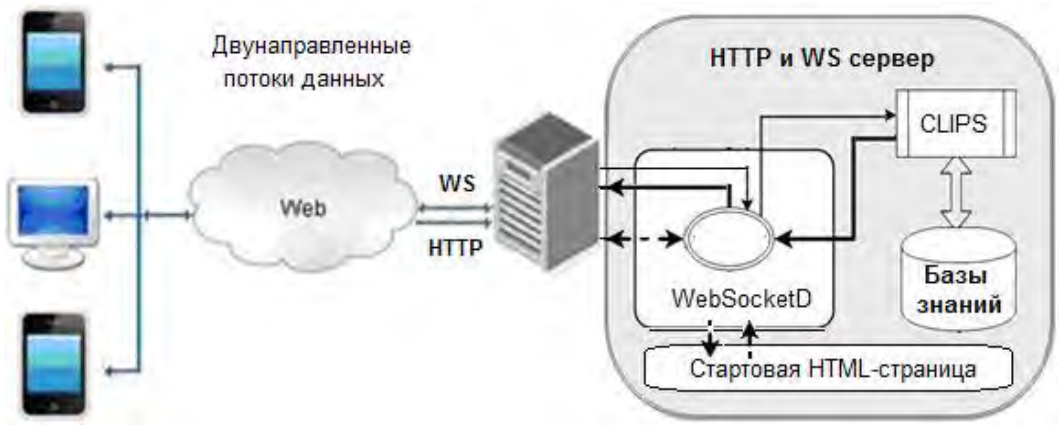


Рис.1 Клиент-серверная архитектура доступу к ядру ЭС

Предполагая, что для связи клиентов с сервером по протоколу websocket будет использован порт 8081, а стартовый index.html сервера находится в текущем каталоге, для запуска сервиса можно использовать команду:

```
websocketd --port=8081 --staticdir=.  CLIPScmd.exe
```

Сервер становится активным и готов к приему HTTP-соединений. Это позволяет в браузере любого клиента ввести HTTP адрес сервера и загрузить в него стартовую html страницу, которая содержит js-код, обеспечивающий возможность работы клиента по протоколу websocket. При этом, обработчик события body.onload() должен содержать следующий фрагмент кода:

```
var url = (location.hostname != '') ? location.hostname : 'localhost';
var ws = new WebSocket('ws://' + url + ':8081/');
```

Это позволяет выполнить переключение клиента на новый протокол и организовать двунаправленный поток данных между клиентом и сервером. Кроме этого, стартовая html страницу должна обеспечивать возможность выбора нужной для консультации базы знаний. Это имеет большое значение, так как настраивает работу CLIPS в составе серверного ядра ЭС.

Так, например, при запуске ЭС в режиме идентификации древесных пород с клиента на сервер должна поступить информации о необходимости загрузки в оболочку CLIPS соответствующего файла базы знаний [5], а также команды на запуск этого файла в режим консультации среды CLIPS. Для рассматриваемого примера это можно осуществить, передав, используя протокол websocket, с клиента на сервер следующую последовательность управляющих команд:

```
(clear)
(load* "Expert_Tree.clp")
(reset)
(run)
```

На сервере поток этих сообщений будет принят утилитой WebSocketD и перенаправлен на вход уже запущенного процесса CLIPScmd, что вызовет очистку его рабочей памяти, загрузку в нее набора фактов и правил из соответствующего файла БЗ, и запуска его в режим новой консультации. Если в этот момент уже выполнялась консультация с каким-либо другим клиентом, то параллельно будет запущен новый процесс.

Запуск на сервере соответствующей БЗ в режим консультации приведет к тому, что среда CLIPS сформирует первый запрос пользователю ЭС. Этот запрос с выходного потока CLIPScmd поступает на вход WebSocketD, где преобразуется в выходной ws-поток сервера. Он, пройдя по сети Интернет, поступит на вход клиента и сформирует новый вид html-страницы (рис. 2).

При этом следует отметить, что вся необходимая для формирования страницы браузера информация, как текстовая, так и графическая передается по сети с использованием протокола websocket, что позволяет существенно сократить время обмена информацией между клиентом и сервером, и как следствие существенно сократить загрузку сети и сервера.

Все, что необходимо сделать пользователю ЭС для получения желаемой консультации – это отвечать на уточняющие вопросы, предлагаемые ЭС, которые формируются базой знаний CLIPS. В случае, который приведен на рис. 2, для ответа на вопрос достаточно щелкнуть мышью в соответствующей области экрана. Это действие будет трансформировано в соответствующий ответ и по протоколу websocket будет возвращено на сервер, где утилита WebSocketD примет его и уже в нужном формате перенаправит на вход CLIPS.

Отработает процедура логического вывода оболочки CLIPS, выполнится следующий этап работы с базой знаний, на основе которого сформируется новый уточняющий вопрос к пользователю ЭС. И так будет продолжаться до тех пор, пока ЭС не получит конечное заключение или распишется в своей беспомощности (рис. 3).

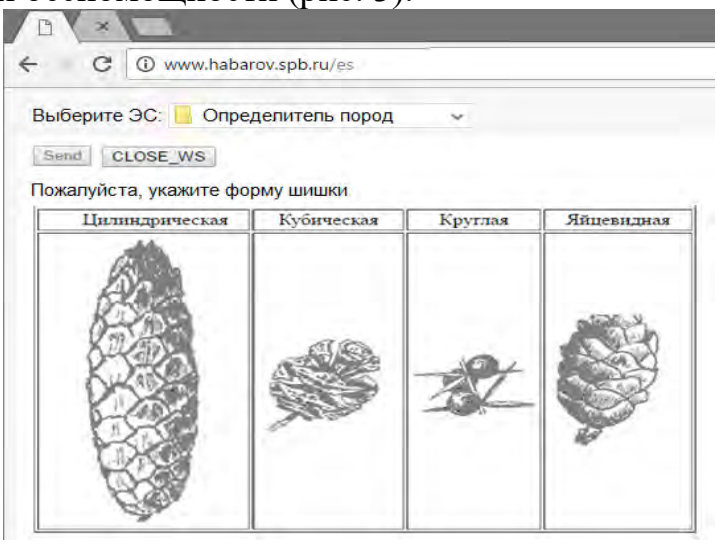


Рис.2 Начало консультации экспертной системы

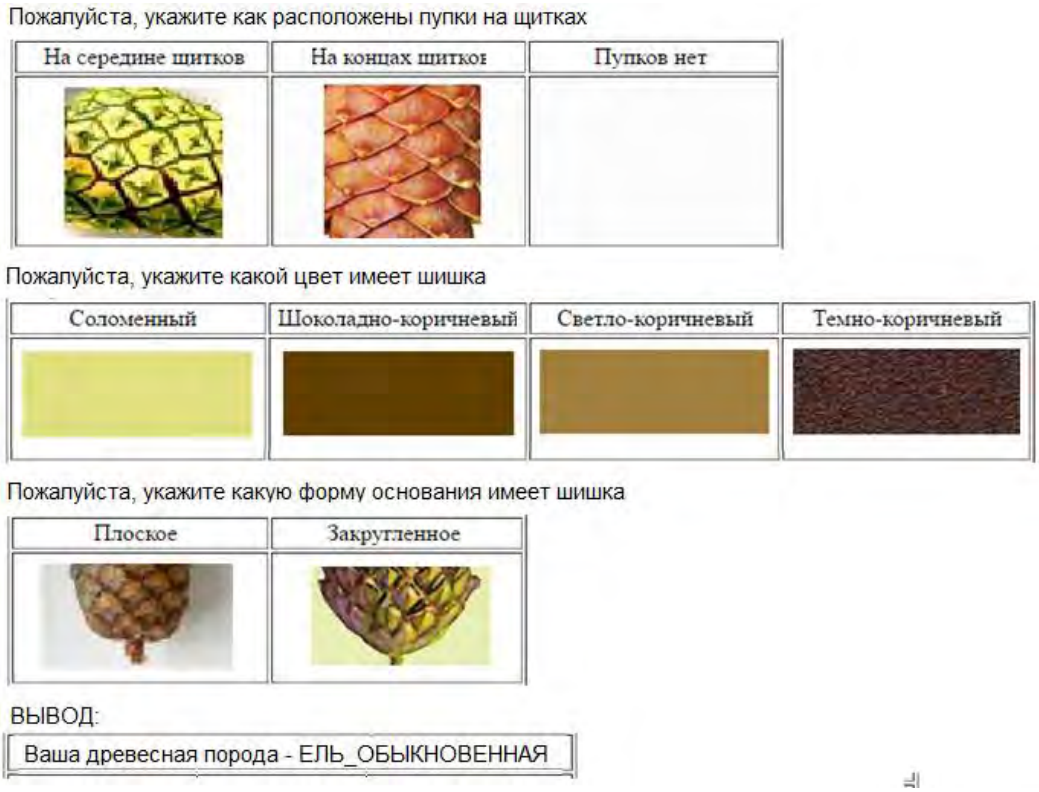


Рис.3 Процесс и результат проведения консультации экспертной системы

Таким образом, используя предложенный подход, пользователю для получения консультации от какой-либо экспертной системы достаточно обладать всего лишь браузером и знать адрес соответствующего сервера. При работе в удаленном режиме с использованием протокола websocket у пользователя будет ощущения, что все это происходит на его компьютере.

Библиографический список

1. Хабаров С.П.. Организация гетерогенных ЛВС с терминальным доступом между ее узлами. //Известия Санкт-Петербургской лесотехнической академии – СПб.:СПбГЛТУ, 2016, Вып. 216, с. 267-280.

2. Заяц А. М., Логачев А. А. Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей. //Известия Санкт-Петербургской лесотехнической академии – СПб.:СПбГЛТУ, 2016, Вып. 216, с. 241-255.

3. Хабаров С.П. Использование утилиты websocketd для удаленного выполнения программ. //Сборник научных трудов "Информационные системы и технологии: теория и практика" – СПб.:СПбГЛТУ, 2017. № 9.

4. Хабаров С.П. Взаимодействия узлов сети по протоколу websocket. //Сборник научных трудов "Информационные системы и технологии: теория и практика" – СПб.:СПбГЛТУ, 2017. № 9.

5. Голубев К.С., Заяц А.М., Хабаров С.П. Интеллектуальная система "Определитель хвойных пород по шишкам". Свидетельство о государственной регистрации программ для ЭВМ № 2015613729 от 24.03.2015.

С.П. Хабаров, кандидат технических наук, доцент
И.А. Красовский, студент 2 курса магистратуры
А.Х.-А. Киев, студент 2 курса магистратуры

УДАЛЕННОЕ УПРАВЛЕНИЕ НА БАЗЕ ТЕХНОЛОГИИ WEBSOCKET

Двунаправленность потоков данных через Интернет и постоянная связь между клиентом и сервером без лишних HTTP заголовков делают протокол WebSocket привлекательным для построения распределенных управляющих систем реального времени. Кроме этого, наличие WebSockets AP позволяет достаточно просто использовать его в web-приложениях для открытия или закрытия соединения, передачи и получении я сообщений [1,2]. И что особо важно, WebSocket поддерживают все современные браузеры разных ОС [3].

В данной статьи исследуется возможность технологии WebSocket для построения распределенных управляющих систем. С этой целью рассмотрен пример простейшей системы управления всего двумя объектами, на которые подаются сигналы управления и считываются сигналы их состояния (рис. 1).

Рис.1. Структура рассматриваемого примера системы

При этом система должна позволять любому из удаленных клиентов, используя только браузер, просматривать текущее состояние всех объектов управление. А клиенты, имеющие соответствующие права, могли бы еще, и управлять доступными им объектами. Без потери общности предположим, что речь идет об управление подачей электроэнергии в два разных дома.

Для простоты реализации примера принимается его клиент-серверная организация, при которой состояние всех объектов хранится на сервере, а их количество, в общем случае, может быть значительно больше, чем два. При этом каждый из клиентов должен поддерживать только одно ws-соединение с сервером. При построении чисто распределенной системы без сервера всем клиентам необходимо было бы организовывать и поддерживать соединения со всеми объектами, что значительно усложнило

рассматриваемый пример. Оставаясь в рамках клиент-серверной парадигмы, отметим, что:

- Серверная часть приложения будет использовать кроссплатформенную утилиту WebSocketD, которая выполняет функции WebSocket-сервера с перенаправление внешних входных и выходных потоков сервера на серверный код обработки.
- Серверный код обработки будет реализован средствами объектно-ориентированного языка программирования JAVA.
- Клиентская часть приложения будет представлять собой простую HTML-страницу, которая с помощью WebSocket-подключения будет обращаться к серверу и обмениваться с ним сообщениями.

Реализация серверной части приложения. При создании серверного кода необходимо предусмотреть возможность *многопоточности*. То есть серверная часть должна отслеживать все изменения на каждом из клиентов, обрабатывать их и отправлять изменения состояние системы всем клиентам подключенным к серверу в текущий момент. Потоки взаимодействовать друг с другом через основной «родительский» поток, из которого они стартовали.

Для реализации поставленной цели создадим файл с расширением *.java, в котором определим два класса: *LampServer* (рис. 2) и *FonStream* (рис. 3). Для простоты реализации, информацию о текущем состоянии всех объектов управления будем хранить в файле *lamps.txt*, формат содержимого которого имеет следующий вид:

1 <ON/OFF>|2 <ON/OFF>| … |<n> <ON/OFF> ,

Файл состоит из отдельных, разделенных символом ”|”, записей, первое поле которых определяет номер объект управления, а второе его состояние

```
public class LampServer {
    static String fileName = "lamps.txt";

    public static void main(String[] args) throws IOException  {
        Thread Stream2 = new Thread(new FonStream());  //Создание потока Stream2
        Stream2.start();                               //Запуск потока
        while (true) {
            BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
            String message = in.readLine();
            if (message.length()==0){
                System.exit(0);
            }
            System.out.println(message);
            ReplaceFile(message);
        }
    }
```

Рис. 2. Скриншот кода класса LampServer

Класс *LampServer*, являясь родительским, создает новый поток *Stream2* на основе класса *FonStream*. Дочерний поток запускается (*start*), а основной поток переходит в режим ожидания приема сообщений от клиен-

тов, которые на него перенаправляет утилита WebSocketD. При приходе сообщения оно обрабатывается, выполняется команда на включение или отключение того или иного объекта, и обновляется информация в файле *lamps.txt*.

Классс *FonStream* реализует интерфейс *Runnable* (рис. 3), который. является наиболее распространённым способом создания потоков.

```
class FonStream implements Runnable {
    //метод run будет выполняться в фоновом потоке
    public void run()  {
        try{
            String line0 = LampServer.ReadLineFromFile();
            System.out.println(line0);
            while (true) {
                String line1 = LampServer.ReadLineFromFile();
                if (line1.compareTo(line0)==0) {
                    try{
                        sleep(3000);
                    }catch (InterruptedException e){}
                } else {
                    System.out.println(line1);
                    line0 = line1;
                }
            }
        }catch (IOException e){}
    }
}
```

Рис. 3 Скриншот кода класса FonStream

В данном классе каждые 3 секунды сравнивается содержимое файла lamps.txt с текущим состоянием объектов. При совпадении данных ничего не происходит, а если данные отличаются, то происходит их актуализация и текущее состояние системы будет передано всем клиентам, подключенным в текущий момент к серверу по протоколу WebSocket.

Реализация клиентской части приложения. Она ориентирована на работу в среде стандартных браузеров с поддержкой WebSocket и создается средствами HTML и JavaScript. Общий вид страницы при её первоначальном запуске представлен на рис. 4.

Рис. 4. Скриншот главной страницы приложения

Вызов главной страницы выполняется стандартным указанием http:// адреса сервера, но при ее загрузке произойдет автоматическое подключение клиента к WebSocket-серверу, о чем информирует сообщение в нижней чаи страницы (*Connected to Server*).

Клиент имеет возможность выбрать режим доступа к управлению объектами. Выбор одного из режимов выводит на страницу дополнительные элементы управления (кнопки), которые и предоставляют возможность управлять электроэнергией конкретного объекта (рис. 5).

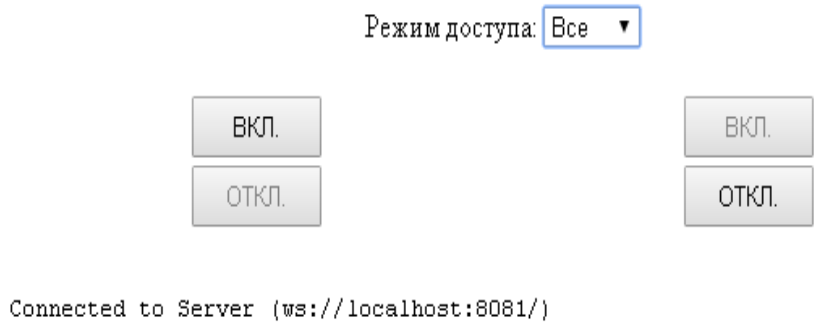


Рис. 5. Скриншот главной страницы с режимом доступа «Все»

У каждого из объектов свои кнопки управления. При нажатии на одну из кнопок «Вкл./Выкл.» на сервер отправляется команда на выполнение нужной операции на конкретном объекте. Формат отправляемого сообщения: ”<номер_объекта> <операция>”. Например, '1 ON’.

Поступившее на сервере сообщение анализируется, и при наличии в нем подстроки ‘ON’ или ‘OFF’ будет выполнена соответствующая актуа-

лизация системы. При поступлении от сервера нового ws-сообщения обработчик события ws.onmessage клиента вызывает функцию для его анализа (рис. 6).

```
function ReciveFromServer(message) {
    block = message.split('|');
    for (var i = 0; i < block.length; i++) {
        msg = block[i].split(' ');
        n = msg[0];
        if (msg[1]=="ON") {
            $('L'+n+'_OFF').style.display = 'none';
            $('L'+n+'_ON').style.display = 'inline';
            $('on'+n).disabled = true;
            $('off'+n).disabled = false;
        } else if (msg[1]=="OFF") {
            $('L'+n+'_OFF').style.display = 'inline';
            $('L'+n+'_ON').style.display = 'none';
            $('on'+n).disabled = false;
            $('off'+n).disabled = true;
        } else {
            log('Error: ' + i + ' - ' + block[i]);
        }
    }
}
```

Рис. 6. Скриншот кода обработки сообщений сервера

Эта функция поступившее сообщение представляет массивом записей для каждого из объектов управления. Анализ этих записей приводит к смене изображений на главной странице, а также регулирует доступность кнопок по управлению каждым из объектов.

Запуск приложения. На приведенной ниже фотографии (рис. 7) показано сетевое тестирование приложения, созданного с использованием технологии WebSocket. Удаленно с устройства «iPad» было произведено подключение по внешнему IP адресу к серверу разработанной системы, после чего был получен доступ к удаленному управлению подачей электроэнергии на каждый из объектов.

Рис. 7. Демонстрация удаленного управления освещением домов

Но для того, чтобы эксперимент закончился успешно, необходимо было предварительно на сервере выполнить ряд операций:

• Загрузить консольную версию утилиты WebSocketD и установить на сервере программную среду Java.
• Используя компилятор Java, превратить исходный код *LampServer.java* в байт-код, соответствующий спецификации JVMS

```
javac LampServer.java
```

• На базе откомпилированных *FonStream.class* и *LampServer.class* создать для серверного кода выполняемый jar-архив:

```
jar -cvmf manifest.mf  LampServer.jar  *.class
```

используя при этом файл манифеста простейшего вида:

```
main-class: LampServer
```

• Для запуска WebSocket-сервера сформировать командный файл вида:

```
Websocketd  --port=8081  --staticdir=.\htm  java  -jar  LampServer.jar
```

Структура командного файла предполагает, что после запуска утилиты WebSocket-сервер будет работать на 8081 порту сервера, все внешние входные и выходные ws-потоки утилита будет транслировать процессу LampServer.jar, который сама автоматически и запустит. Кроме этого, данный командный сценарий предполагает, что все файлы главной html-страницы клиентской части приложения хранятся в папке .\htm сервера.

Выполнив указанные выше действия, теперь для запуска системы в работу достаточно запустить на выполнение командный файл. При этом HTTP и WS-сервера станут доступны в сети.

Заключение. В качестве заключения следует отметить, что технология на базе WebSocket проста в использовании, быстра, надежна и достаточно эффективна. Обеспечивает двунаправленную связь между клиентом и сервером и подходят для тех приложений, которые получают и отсылают много данных за очень короткий промежуток времени в реальном времени. Особую роль эта технология может занять в составе систем, одним из узлов которых являются сегменты сенсорных сетей сбора информации [4] .

Библиографический список

1. Хабаров С.П. Использование утилиты websocketd для удаленного выполнения программ. //Сборник научных трудов "Информационные системы и технологии: теория и практика" – СПб.:СПбГЛТУ, 2017. № 9.
2. Хабаров С.П. Взаимодействия узлов сети по протоколу websocket. //Сборник научных трудов "Информационные системы и технологии: теория и практика" – СПб.:СПбГЛТУ, 2017. № 9.

3. Хабаров С.П.. Организация гетерогенных ЛВС с терминальным доступом между ее узлами. //Известия Санкт-Петербургской лесотехнической академии – СПб.:СПбГЛТУ, 2016, Вып. 216, с. 267-280.

4. Заяц А. М., Логачев А. А. Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей. //Известия Санкт-Петербургской лесотехнической академии – СПб.:СПбГЛТУ, 2016, Вып. 216, с. 241-255.

М.Л. Шилкина, кандидат технических наук, доцент

МИГРАЦИЯ ЛОКАЛЬНОЙ БД В БАЗУ ДАННЫХ MYSQL НА ИНТЕРНЕТ-РЕСУРСЕ

Многие предприятия различных сфер бизнеса − сферы обслуживания, автосервиса, магазины, оптовые склады, − имеют свою локальную или клиент-серверную базу данных, содержащую информацию о клиентах, предоставляемых услугах, о номенклатуре и характеристиках реализуемых товаров и услуг, прайс-листы, информацию о сотрудниках фирмы. С другой стороны, любое предприятие в соответствии с требованиями современных реалий бизнеса поддерживает свои собственные Интернет-ресурсы, где потенциальные клиенты могут получить рекламную информацию о товарах и услугах фирмы. На определенном этапе развития предприятия возникает необходимость не только информировать клиента (или сотрудника) о своём бизнесе, но и вести сам бизнес через Интернет − посредством открытия Интернет-магазина, бронирования мест в гостиницах через глобальную сеть и т.п. В значительном большинстве случаев Интернет-бизнес возникает не сам по себе, а на основе уже существующего бизнес-предприятия с офисом и стабильным доходом, например, обычного магазина, склада, гостиницы и т.п., потому что, как показывает практический опыт, доход от интернет-продаж существенно ниже, чем от офисных продаж. Возникает проблема ведения двойного бизнеса − для продаж через офис и через интернет. Чтобы избежать её, требуется объединить базы данных обеих видов продаж в одну согласованную базу. Задача осложняется тем, что локальная база данных, как правило, создана в формате Access, MS SQL Server, Oracle, Paradox и т.д., а база данных на интернет-ресурсе имеет, как правило, формат MySQL. Можно было бы экспортировать содержание локальной базы в формат XML, а затем, используя PHP-библиотеку SimpleXML, импортировать содержимое XML-файла в базу данных MySQL, но этот сценарий мало пригоден с практической точки зрения, так как продажами занимаются менеджеры, которые не компетентны для работы с базами данных на Web-серверах. Поэтому требуется максимально автоматизировать процедуру согласования состояний локальной и Web баз данных.

Таким образом, постановка задачи формулируется так: имеется клиент-серверная или локальная база данных частного предприятия и сайт этой же фирмы, разработанный в CMS и размещенный на хостинге в интернете, со своей базой данных MySQL. Требуется разработать механизм автоматизированного изменения контента на сайте при изменении соответствующего содержания в базе данных предприятия.

Эту задачу предлагается решить с использованием автоматически запускаемого на Web-сервере скрипта, написанного на языке Perl, который по заданному расписанию проверяет состояние папки для входных файлов. Если скрипт обнаруживает в этой папке XML-файл для обновления состояния таблиц базы данных MySQL, скрипт сначала разбивает этот входной файл на множество файлов фиксированного размера, причём необходимость такого разбиения вызвана тем, что на хостингах стоят ограничения на размер входных файлов для последующей обработки их функциями библиотеки SimpleXML. Полученные в результате разбиения файлы скрипт сохраняет во временную папку. Затем из каждого файла скрипт поблочно извлекает значения параметров, которые подставляет в хранимую процедуру, модифицирующую состояние базы данных на Web-ресурсе, и запускает эту процедуру на выполнение (рис. 1).

Для решения поставленной задачи необходимо определенное аппаратное и программное обеспечение. Если отлаживать задачу надо на локальном сервере, то сначала необходимо установить этот локальный сервер, в состав которого входят:

1. Apache (2.4 и старше);
2. PHP (5.5 и старше);
3. MySQL – желательно последняя версия;
4. графическая оболочка phpmyadmin для работы в MySQL (по желанию).

Для упрощения настройки совместной работы этих компонентов можно установить готовый локальный сервер Wamp, EasyPHP или Denver. Кроме того, необходимо установить на локальном компьютере графическую среду Pedro для разработки и отладки Perl-скриптов на локальном сервере. Вывод диагностических сообщений Perl осуществляет на консоль, в cmd-строку.

Если решать поставленную задачу надо на арендованном для размещения сайта (и, соответственно, базы данных MySQL) сервере на хостинге, то хостинг должен также предоставлять Apache не ниже 2.4, PHP 5.X, MySQL и Crontab для запуска Perl-скриптов.

Рассмотрим решение поставленной задачи (рис. 1) на примере. На практике возможна ситуация, когда на сайте уже установлен стандартный компонент, который, например, реализует Интернет-магазин, вроде VirtueMart в CMS Joomla. В этом случае структура входного XML-файла должна определяться структурой таблиц этого компонента.

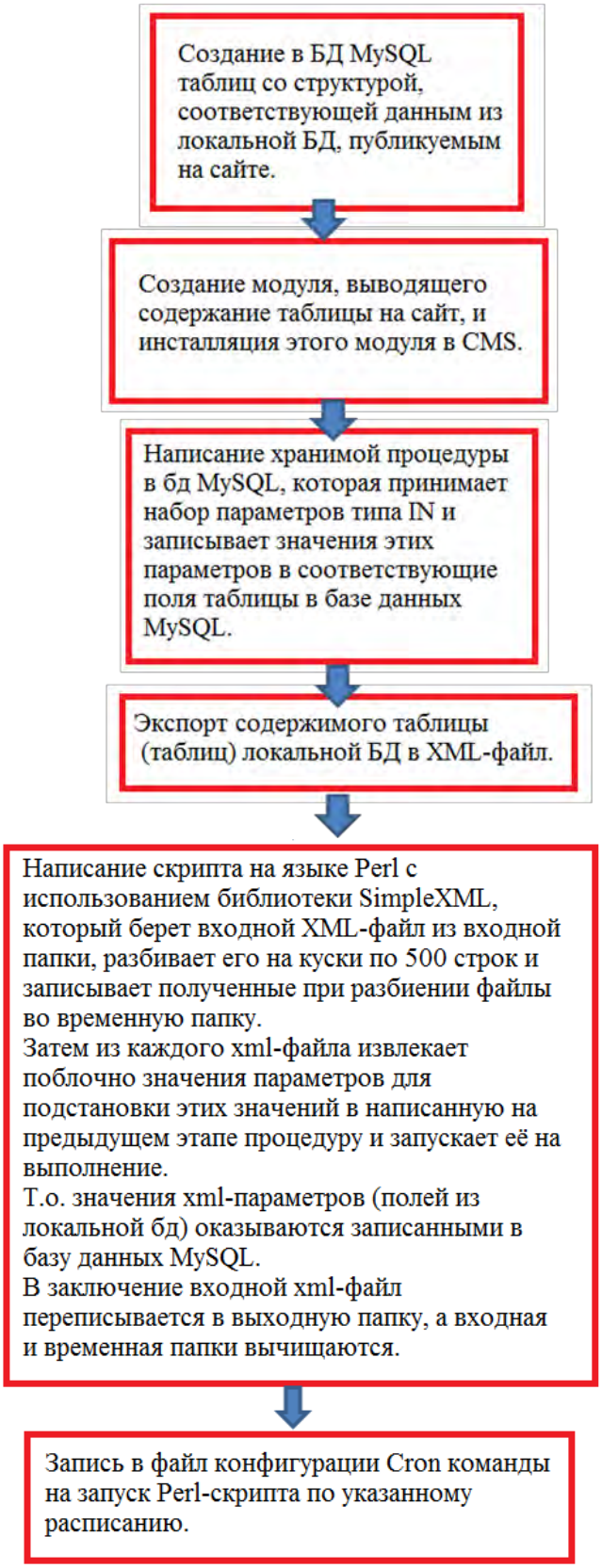


Рис. 1. Схема согласования состояний локальной базы данных и базы данных на Web-ресурсе

С другой стороны, возможна ситуация, когда на сайте требуется отобразить текущее состояние некоторых таблиц локальной базы данных. Тогда для входного XML-файла определяющей является структура именно

этих таблиц. Рассмотрим этот случай постановки задачи. Так как отображать на сайте надо таблицу из локальной базы, то на сайте нет готового компонента для вывода содержимого таблицы, и нет изначально в базе данных MySQL таблицы, соответствующей таблице (или таблицам) локальной базы данных. Поэтому первое, что необходимо сделать - это создать в MySQL таблицу с полями, соответствующими данным из локальной базы данных, которые необходимо вывести на сайт.

Это можно сделать двумя способами:

a. из phpmyadmin через графический интерфейс;
b. через директиву CREATE TABLE из командной строки MySQL.

Пусть в таблице, которую надо опубликовать, 5 полей:

- id INT(11);
- name VARCHAR(20);
- text1 text;
- quant INT(11);
- data DATE.

После того, как таблица в MySQL создана, надо создать модуль вывода содержимого этой таблицы на сайт. Все компоненты модуля стандартные как для любого модуля соответствующей CMS, но в **default.php** должен присутствовать блок, содержащий запрос к базе MySQL на данные, которые надо выводить на страницу:

```
$db = JFactory::getDBO();
$query  = " SELECT * FROM  `имя_таблицы_MySQL` ";
$db->setQuery($query);
      $list = $db->loadObjectList();
      echo  '<p><table  class="имя_класса_CSS"  border="5"
                  cellpadding="5" cellspacing="1" >
            <tr align="center" >
                <th>id</th>
                <th>Имя</th>
                <th>Order</th>
                <th>Cost</th>
                <th>Data</th>
            </tr>';
foreach ($list as $item) {
  echo '
  <tr   align="left">
      <td>  '.$item->id.'
      <td>  '.$item->name.'
      <td>  '.$item->text.'
      <td>  '.$item->quant.'
```

```
            <td>  '.$item->data.'
        </tr>';
    }
    echo '</table></p>';
```

Следует сделать несколько замечаний по поводу кодировок таблиц в MySQL:

1. Файлы с описанием модуля и входной XML-файл с импортируемым содержимым должны быть сохранены в кодировке UTF-8, иначе в браузере при просмотре страницы с модулем, выводящим таблицу, и в командной строке Perl появятся ошибки.

Это справедливо для случая, если есть необходимость вывода кириллицы. Для латиницы модуль может корректно работать и в других кодировках.

2. Кодировка БД MySQL, а также сравнение строк в таблице, в которую импортируются данные, должны быть установлены utf8_general_ci или cp1251_general_ci, иначе русский шрифт будет отображаться иероглифами.

Когда модуль вывода таблицы написан и корректно работает, можно приступать к написанию хранимой процедуры, заполняющей выводимую таблицу нужными значениями полей и получающей в качестве параметров значения этих полей:

```
    DROP PROCEDURE `имя_процедуры`
    CREATE DEFINER=`user_name`@` host_name` PROCEDURE ` имя_процедуры `(
            IN Person_Id  int(11),
        IN Name  varchar(20),
        IN Text1  text,
        IN Quantity  int(11),
        IN Data  date)
    BEGIN
    IF NOT EXISTS(SELECT * from имя_таблицы_MySQL where id = Person_Id)
    THEN
      INSERT into имя_таблицы_MySQL (id, name, text1, quant, data)
      values(Person_Id, Name, Text1, Quantity, Data );
      ELSE
      UPDATE имя_таблицы_MySQL set
        name = Name, text1 = Text1, quant = Quantity, data = Data
      Where id = Person_Id;
    END IF
    END
```

Следующим этапом является экспорт содержимого локальной базы данных в XML-файл. Для создания XML-файла из таблицы в Access надо:

1. Правым щелчком мыши по имени таблицы вызвать контекстное меню.

2. Выбрать *Экспорт-> XML-файл*

3. Через кнопку Обзор указать папку назначения и имя создаваемого XML-файла.

Переименовываем в XML-файле:

– root-тег в *Container*,

– построчные табличные теги в *Procedure_SQL* с атрибутом *Name= "имя_хранимой_процедуры ",*

– теги значений полей кортежей, вложенные в построчные теги, переименовываем в *Param,* важно, чтобы порядок следования этих вложенных тегов соответствовал порядку следования формальных параметров хранимой процедуры.

Теперь можно приступать к созданию скрипта на языке Perl. Для его корректной работы сначала надо создать 3 папки на сервере (или на локальном компьютере, если работа осуществляется на localhost) с именами:

- xml - папка для загрузки туда входного xml-файла;
- xml_tmp - временная папка для записи туда разбитого на куски входного XML-файла;
- xml_old - папка для записи туда обработанного входного xml-файла.

В первую папку сохраняем подготовленный XML-файл, остальные папки пусты.

Скрипт содержит следующие функциональные блоки:

1. Проверяет папку XML на предмет наличия в ней файлов.
2. -> Если файл есть
3. - очищает соответствующую таблицу с данными в БД MySQL;
4. - разбивает файл на куски входной XML-файл и записывает эти куски в папку xml_tmp;
5. - читает структуру каждого XML-блока в память;
6. - запускает написанную хранимую процедуру с параметрами на SQL-сервере;
7. - копирует отработанный XML-файл в папку xml_old.

Главное тело скрипта имеет вид:

```
use Data::Dumper;       # выполняет print на консоль Perl
use XML::Simple;
use DBI;                # устанавливает соединение с БД
use File::Copy;
use Encode;
my $dir      = "/путь_к_папке_xml/xml";
my $dir_end  = "/путь_к_папке_xml_old/xml_old";
my $dir_tmp  = "/путь_к_папке_xml_tmp/xml_tmp/";
my @list_dir = scan_dir($dir); #  записывает имена файлов из папки xml в
 #   массив list_dir
```

```
my @list_tmpdir;
my $num;   my $numm;
my $text;    my $text1;
my $data_file;
my $data_file_tmp;
my @params;
my $sql3;
my $sth;
my $rv;
my $user='user_name';
my $password='pass';
my $database = 'database_name';
my $host = 'host_name';
my $sql;
my  $dbh  =  DBI->connect("DBI:mysql:$database:$host",$user,  $pass-
word);
                    # подключились к БД MySQL
foreach $data_file (@list_dir) {
$sql3 = "TRUNCATE TABLE имя_таблицы_MySQL";
        $sth = $dbh->prepare($sql3);
        $rv = $sth->execute;
      clean_dir($dir_tmp); #  очистили папку xml_tmp
      wrap_file($data_file); #    разбили исходный xml-файл на куски и
                  #   записали их в xml_tmp
      @list_tmpdir = scan_dir($dir_tmp);
    # записали в массив list_tmpdir имена файлов из папки xml_tmp
      foreach $data_file_tmp (@list_tmpdir){
            my $simple = XML::Simple->new();
            my  $data  =  $simple->XMLin(  $data_file_tmp,  ForceArray
=> 1);
        #  записывает вложенные теги в массивы
            foreach $num ( @{ $data->{Procedure_SQL} }) { $#params
= -1;
        #  очистили массив params
                  $text1 = $num->{Name};
                  $sql ="call " . $text1 . "(";
                  foreach $numm ( @{ $num->{Param} } ) {
                        $text = $numm->{content};
                        push @params, $text;
                        $sql = $sql . " '" . $text . "',";
                  }
                  chop($sql);
                  $sql = $sql . ")";
                  my $sql2 = encode('cp1251',$sql);
                              $sth = $dbh->prepare($sql2);
```

```
                        $rv = $sth->execute;
                }
        }
                copy($data_file, $dir_end);
                unlink $data_file or warn "fail on $data_file: $!\n";

}
my $rc = $dbh->disconnect;
clean_dir($dir_tmp);
```

В скрипте использовались вызовы вспомогательных процедур. Одна из них формирует массив из имен файлов, находящихся в директории с именем, переданном в процедуру в качестве фактического параметра:

```
sub scan_dir {
my ($dir) = @_;
# массив @_ содержит параметры, передаваемые в процедуру
my @list;
for my $file ( glob( $dir . '/*' ) ) {
# glob() возвращает список файлов, * - любая последовательность символов
        push @list, $file;
        if ( -d $file )   # проверка, что $file является директорием
        {
                my @sub = scan_dir($file); # рекурсивный вызов
                push @list, @sub;
        }
}
return @list;
}
```

Другая процедура удаляет из папки все содержимое:

```
sub clean_dir{
my @clean_dir = scan_dir($_[0]); # $_[0] - первый передаваемый параметр
foreach $clean_file (@clean_dir){
     unlink $clean_file or warn "fail on $clean_file: $!\n";
     # unlink удаляет файл
}
}
```

Последняя пользовательская процедура разбивает файл, переданный в нее в качестве входного параметра на файлы фиксированного размера по 500 строк.

```
sub wrap_file {
my ($file) = $_[0];
my @list;
```

```
my $i = 0;
my $j = 0;
my $fileN = "$file"."_n";
    open(INPUT, "<", $file) or die "Couldn't open $file \n" ;
    open(OUTPUT, "+>", $fileN) or die "Couldn't open $fileN \n" ;
  # создаём файл для чтения и записи одновременно
    while (<INPUT>){
        s/<\/Procedure_SQL>/<\/Procedure_SQL>\n/g;
     # подставляет везде после закрывающих тегов перевод строки
        print OUTPUT "$_";
     # записываем изменённые строки в OUTPUT
    }
    close(INPUT);
    close(OUTPUT);
    open(INPUT, "<", $fileN) or die "Couldn't open1 $file \n" ;
  # открываем INPUT в read - mode
    open(OUTPUT, "+>", "$dir_tmp$j") or die "Cannot open
$dir_tmp$j ";
    while (<INPUT>) {
        if   (/<\/Procedure_SQL>$/) {
     # если строка  заканчивается на  закрывающий  тег
        print OUTPUT "$_"; # переписываем i-ю строку в OUT-
PUT
            $i++;
            if ($i>500){
                $i = 0;
                $j++;
                print OUTPUT "<\/Container>";
                close(OUTPUT);
                open(OUTPUT, "+>", "$dir_tmp$j") or die
                  "Couldn't open3 \n" ;
              print OUTPUT "<?xml version=\"1.0\"
        encoding=\"Windows-1251\" standalone=\"yes\"?>";
                print OUTPUT "<Container>";
            }
        }
        else { print OUTPUT "$_" }
    }
    if ((i <= 500) and (j>0))
      { print OUTPUT "<\/Container>"; }
    close(OUTPUT);
    push @list, $file;
    close(INPUT);
    unlink $fileN or warn "failed on $fileN with break $!\n";
    return @list;
```

}

Теперь, когда скрипт написан, необходимо задать расписание, по которому программа Cron будет запускать его на выполнение. Cron − это программа, выполняющая задания по расписанию. Позволяет неоднократный запуск заданий. Т.е. задание можно запустить в определенное время или через определенный промежуток времени.

На хостинге, как правило, есть интерфейс для работы с Crontab-панелью. Он включает:

– маску расписания запуска скрипта, например, */15 */2 * * * - запускать каждые 15 мин по четным часам, каждый день, месяц и день недели;

– указание пути к интепретатору Perl и через пробел указание полного пути к скрипту.

В заключение необходимо сделать вывод, почему именно таким образом пришлось решать задачу экспорта данных из клиент-серверной базы данных бизнес-предприятия в Web-базу данных MySQL. Ведь, как уже отмечалось, в PHP также есть библиотека SimpleXML, которая функционально аналогична такой же библиотеке в Perl. Проблема в том, что с учетом объёмов современного бизнеса, входные XML-файлы оказываются огромных размеров, а на хостингах стоит ограничение по памяти для исполняемых скриптов. Так что с текущей технологией большой XML-файл в память загружать не получается. Поэтому необходимо разбивать XML-файл на куски, а это классическая задача работы с текстами, для которой создавался Perl. И поэтому силами PHP не обойтись. Немаловажным преимуществом является также возможность автоматического запуска скрипта по расписанию, без необходимости привлечения к этому сотрудников предприятия.

М.А. Шубина кандидат.технических наук, доцент

КОМПЛЕКСНЫЕ РЕШЕНИЯ КАТАЛОГИЗАЦИИ И ХРАНЕНИЯ ДИСТАНЦИОННЫХ ДАННЫХ

С каждым годом расширяется сфера использования данных дистанционного зондирования в различных областях человеческой деятельности. В космосе находится множество аппаратов, фиксирующих различные характеристики поверхности Земли и других планет. Получаемые данные – это данные практически вечного хранения, объем которых постоянно увеличивается.

Космические аппараты принадлежат разным странам, фиксируют разномасштабную информацию в нескольких электромагнитных диапазонах часто об одних и тех же участках Земли. В связи с этим возникают задачи хранения и каталогизации данных, а также доступа с учетом конфи-

денциальности, выборки и их анализа пользователями – организациями и гражданами разных стран.

Принципиально инфраструктура распределенных хранилищ спутниковых данных должна представлять собой интегрированную распределенную среду неоднородных информационных ресурсов – результатов исследования Земли из космоса с представлением данных и каталогов данных в согласованных стандартных форматах, с помощью стандартного интерфейса с использованием сопоставимых технологий выборки и анализа [1, 3, 4].

Решение перечисленных задач весьма сложно, но для некоторых глобальных проектов были найдены практические решения.

К числу таких проектов относится построение «Виртуального Научного Центра данных», который реализовал Международный консорциум SPASE – Space Physics Archive Search and Extract (создан научным сообществом Европы и National Science Foundation, США) [2, 15]. Далее от виртуальных центров научных данных был осуществлен переход к информационным инфраструктурам: Spatial Data Infrastructure (SDI), National Data Infrastructure (NSDI), Global Data Infrastructure (GSDI).

Другой проект – в области изучения планет создан NASA (National Aeronautics and Space Administration) – проект PDS (The Planetary Data System) [14]. Образован международный консорциум и сформирована система on-line архивов, состоящая из нескольких географически распределенных центров, связанных сетью Интернет.

В России развитие геоинформационной инфраструктуры распределенных спутниковых данных выполнялось в проекте Института космических исследований РАН «Распределенная среда интегрированных информационных ресурсов в области космического мониторинга природной среды» (Виртуальный Научный Центр спутниковых данных). В 2003 г. Европейская Комиссия приняла решение о создании INSPIRE (Infrastructure for Spatial Information in Europe) – Глобальной геоинформационной инфраструктуры данных в Европе. Аналогичный проект в Канаде – Canadian Geospatial Data Infrastructure (CGDI).

Основные задачи программы развития геоинформационной инфраструктуры Spatial Data Infrastructure (SDI) можно сформулировать следующим образом:

• Построение глобальной инфраструктуры ИТ и геоданных.

• Гармонизация спутниковой информации; использование согласованного набора стандартов, «понятных» всем участникам данной системы.

• Интероперабельность между независимо созданными приложениями и базами данных; необходимо совместить между собой их интерфейсы, протоколы и форматы данных.

• Поддержка единой политики данных: а) доступ к данным и б) создание и поддержка спутниковой информации – для задач экологии, но открытой и для сельскохозяйственных и транспортных проблем.

Проблемы и особенности, с которыми сталкиваются национальные научные сообщества исследователей Земли из космоса при поиске оптимального доступа к спутниковых данным, были учтены в проекте INTAS IRIS Project, в российской реализации - Integration of Russian Satellite Data Information Resources with the Global Network of Earth Observation Information Systems. Проблема интеграции спутниковых архивов является примером инфраструктурных решений при организации удаленного доступа и распределенной информационной среды [5].

Основное внимание при решении задач поиска ресурсов, взаимодействия с ресурсами, извлечения ресурсов, совместного использования ресурсов уделяется проблемам разработки масштабной информационной инфраструктуры интеграции ресурсов, удаленного доступа и управления гетерогенными ресурсами спутниковых архивов. С этой целью была выполнена реализация информационных систем трех поколений – от проекта первого поколения INTAS IRIS, до GMES HMA. Проект IRIS поддерживает объединение данных регионального спутникового экологического мониторинга.

В рамках проекта обеспечивается доступ российских пользователей к европейской информационной системе INFEO (метаданные / on-line каталоги). При запросе в единой точке системы поиск данных осуществляется одновременно в ресурсах распределенных мировых каталогов. Запрос на метаданные оформляется единым образом для всех спутниковых центров, и информация ищется сразу по всем центрам, доступным из глобальных поисковых систем (в первую очередь, это система EOSDIS/ USA и система INFEO) [17]. При формировании геоинформационной инфраструктуры спутниковых данных решаются задачи использования GIS и Web технологий для разработки новых средств поиска и выбора данных аэрокосмического зондирования; включение российских спутниковых каталогов в международную систему INFEO; создание описания российских спутниковых коллекций и регистрации их в IDN (International Directory Network). При создании единой информационной системы, основанной на концепции интероперабельных распределенных архивов необходимо обеспечить требования действующих международных стандартов: ISO/TC 211, FGDC USA (Federal Geographic Data Committee), CEOS (рис.1) [8].

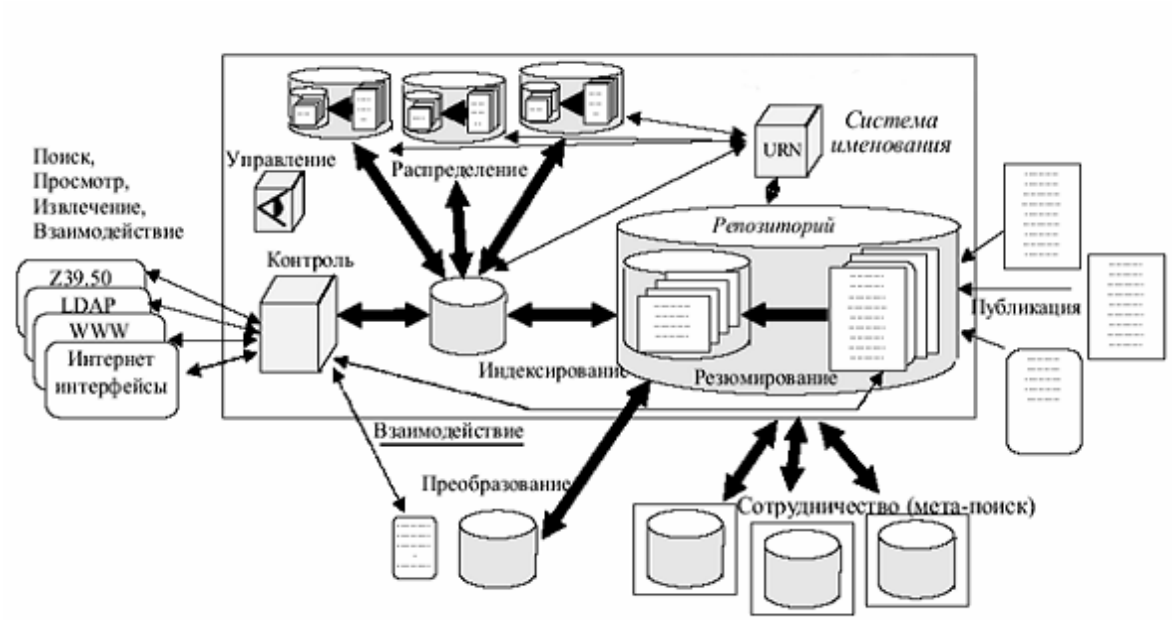


Рис.1. Схема функционирования Архива спутниковых данных в России

Развитие технологий интеграции второго поколения было направлено на переход от подхода, основанного на технологии Z39.50/Gateways, к использованию веб-сервисов (XML над SOAP/HTTP) для связи внешних каталогов с каталогами INFEO [16, 17]. Технологии интеграции второго поколения, получившие название EOLI-XML, разрабатываются ESA в сотрудничестве с национальными космическими Агентствами: Франции (Centre National d'Etudes Spatiales – CNES), Италии (Italian Space Agency – ASI), Германии (German Space Agency – DLR), European Union Satellite Centre (EUCS), Joint Research Centre (JRC) при участии ряда партнеров в промышленности: SPOT Image, Siemens, Spacebel и др. EOLI - XML представляет собой внешний интерфейс для обмена сообщениями между сервером и клиентом на основе протокола SOAP. Технология EOLI-XML лежит в основе системы SSE (Service Support Environment), разрабатываемой для решения указанных задач с помощью создания открытой, сервис - ориентированной, распределенной среды. При дальнейшем развитии выполнялась интеграция неоднородных распределенных информационных ресурсов спутниковых данных (инициатива GMES и проект HMA).

Основным предназначением Global Monitoring for Environment Security (GMES) является поддержка задач Европейского Союза, касающихся устойчивого развития и глобального управления с помощью обеспечения доступа к высококачественным данным дистанционного зондирования Земли в режиме реального времени. Информация, получаемая с помощью проекта GMES, используется в трех основных направлениях деятельности: экология и защита окружающей среды, поддержка развития инфраструктуры ЕС, обеспечение безопасности граждан, как в повседневной жизни, так и в режиме чрезвычайных ситуаций. В настоящий момент доступ к данным дистанционного зондирования Земли каждого спутникового оператора осуществляется через отдельную точку доступа («портал») этого

оператора. Для поиска нужных ему данных пользователь должен просматривать информацию разных операторов отдельно в каждой точке доступа.

С этой целью пользователь может воспользоваться поисковыми интерфейсами, например, по адресу: http://catalogues.eoportal.org/eoli.html, интерфейс которого выполнен в виде Java-апплета. Существует возможность ограничивать поиск по поставщикам данных (панель Find Products), куда входят крупные поставщики, имеющие несколько собственных обширных коллекций и по коллекциям данных (Find Collections).

В настоящее время решением задач в этой области в российском секторе занимается несколько организаций. К числу основных относится ИКИ РАН РФ и сеть НИЦ «Планета», состоящая из подразделений, находящихся в Москве, Новосибирске, Томске, Владивостоке [18]. Участниками проекта являются свыше двадцати институтов РАН, решающих задачи в сфере распределенных вычислений и телекоммуникаций, геофизической, химической, картографической направленности. На протяжении ряда лет совместными усилиями институтов СО РАН, вузов сибирского региона и центров НИЦ “Планета” создается архив данных дистанционного зондирования Земли. Архив, а также его аппаратное и программное обеспечение, система взаимодействия с центрами приема данных составляют ядро информационно-вычис-лительной инфраструктуры Центра коллективного пользования данными дистанционного зондирования СО РАН (рис.2) [8].

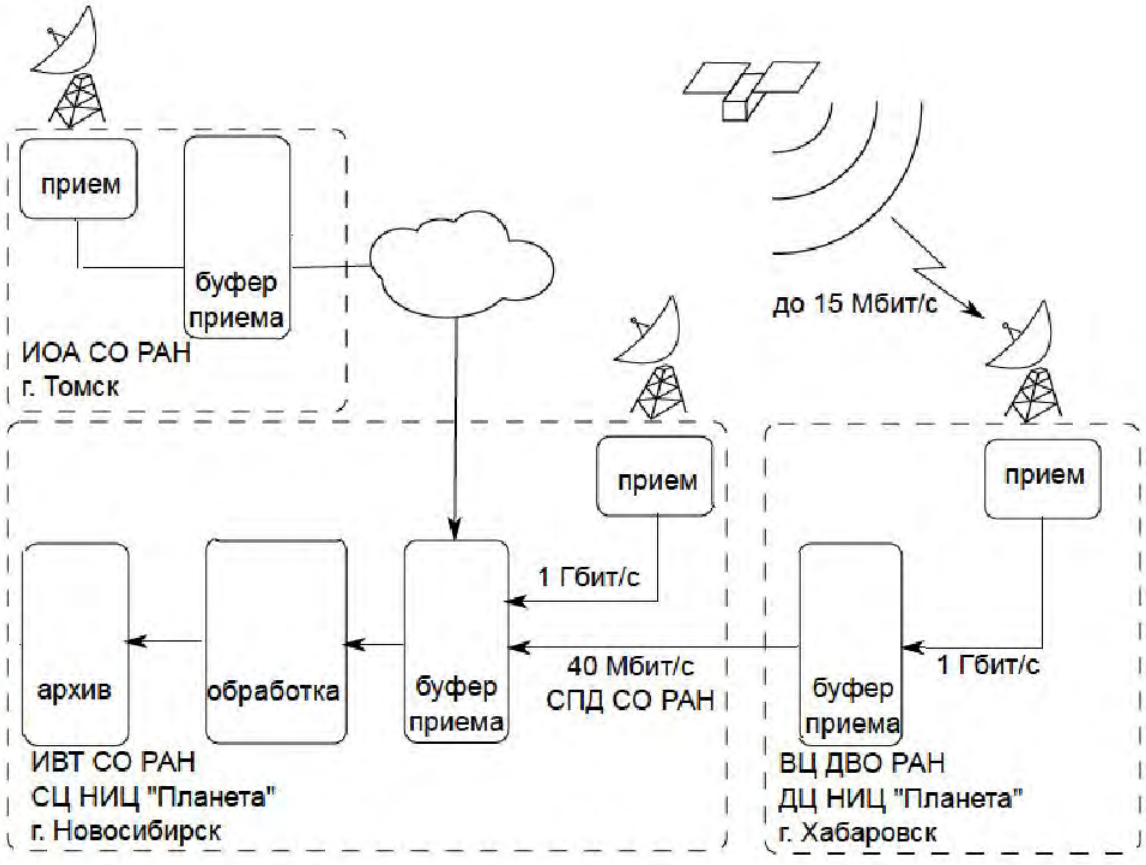


Рис. 2. Распределенная инфраструктура приема, обработки и хранения данных НИЦ «Планета»

Интерфейс НИЦ «Планета» включает поля: «наземный комплекс», «оперативная продукция» по подразделениям.

Продукция Европейского центра ФГБУ "НИЦ "Планета": «обзорные наблюдения, облачность (московский регион, европейский регион, южный федеральный округ, северо-западный регион); «глобальные наблюдения» (карты нефанализа и прогноза эволюции облачных образований; карты метеорологических параметров, обзорные карты снежного и ледового покровов, средняя и максимальная интенсивность и тип осадков осадков, фазы осадков, максимальная скорость восходящих движений, температура поверхности суши и льда, максимальная высота кучево-дождевой облачности, км, возможный диаметр града, интенсивности гроз); «специализированные карты спутникового метеоанализа» (специализированные карты спутникового мониторинга Черного и Азовского морей, карты траекторий движения тропических циклонов); «температура водной поверхности» (Белое море, Балтийское море, Баренцево море, Черное и Азовское моря, Каспийское море, Тихий океан, Индийский океан, Атлантический океан, мировой океан (Единая система информации об обстановке в Мировом океане (ЕСИМО); «ледяной покров»; «снежный покров», «растительный покров», «пожарная обстановка», «паводковая обстановка», «вулканическая активность».

Продукция Сибирского центра ФГБУ "НИЦ "Планета": «обзорные наблюдения, облачность» (сибирского региона), «карты нефанализа и прогноза эволюции облачных образований», «ледяной покров», «ледяной покров на реках и водохранилищах», «пожарная обстановка», «снежный покров», «паводковая обстановка».

Продукция Дальневосточного центра ФГБУ "НИЦ "Планета" включает аналогичные данные для дальневосточного региона и, конечно, данные о вулканической деятельности.

Прикладные проекты: космический мониторинг загрязнения прибрежных акваторий Азово - Черноморского бассейна; технологии космического мониторинга ледяных покровов Арктики и Антарктики; спутниковый мониторинг затоплений Волго-Ахтубинской поймы; мониторинг опустынивания в Калмыкии; оценка воздействия нефтяных и газовых разработок на окружающую среду; международный проект "ICEWATCH"; мониторинг Черного и Азовского морей и др.

Кроме того, выпускаются ежемесячные и итоговые бюллетени и декадные обзоры.

ИКИ развивает методы и технологии «Электронные Библиотеки спутниковых данных изменения окружающей среды и климата», которые особенно важны для развития информационной поддержки космического мониторинга объектов природной среды и глобальных изменений климата, создания единой системы автоматической обработки информации и распределенного поиска по всем национальным архивам космической информации. Электронная Библиотека создается как инфраструктура системы международного обмена космической информацией на основе технологий

INFEO (Information about Earth Observation). Электронная Библиотека ИКИ предоставляет пользователям специализированные интерфейсы доступа к данным, обеспечивает долговременное хранение данных и позволяет производить поиск спутниковых снимков и коллекций данных.

Тем не менее, остается актуальной основная проблема: отстают адекватные способы выбора и анализа больших архивов накопленных данных для прикладных пользователей.

Одним из способов решения проблемы – построение интеллектуальных интерфейсов, включающих элементы обработки данных. Примером может служить разработка технологии GEOSMIS – построения веб-интерфейсов работы с пространственной информацией в системах дистанционного мониторинга, которые предназначены для работы с большими распределенными многомерными архивами спутниковых данных и результатами их обработки. Технология рассчитана не только на поиск данных, но и на создание инструментов анализа и управления данными [7]. В дальнейшем на основе GEOSMIS разрабатываются специализированные интерфейсы для прикладных задач (состояния сельскохозяйственных земель, лесопокрытых территорий и т.д.).

GEOSMIS состоит из двух уровней: уровня представления, включающего просмотр в ArcGis, экспорт данных в GoogleEarth, картографический интерфейс, комбинированный интерфейс, и прикладного уровня, включающего системные модули API и Web-сервисов. Модули API: формирование метаданных, доступ к данным, контроль доступа, контроль процессов. Сервисы: картографические, сервисы метаданных, управление данными, CGI модули. ***Web-сервисы*** позволяют получить доступ к данным на основе протоколов HTTP. Архитектура веб-интерфейса приведена на рис. 3, работа с с ***Web-сервисами представлена на рис. 4 [7].***

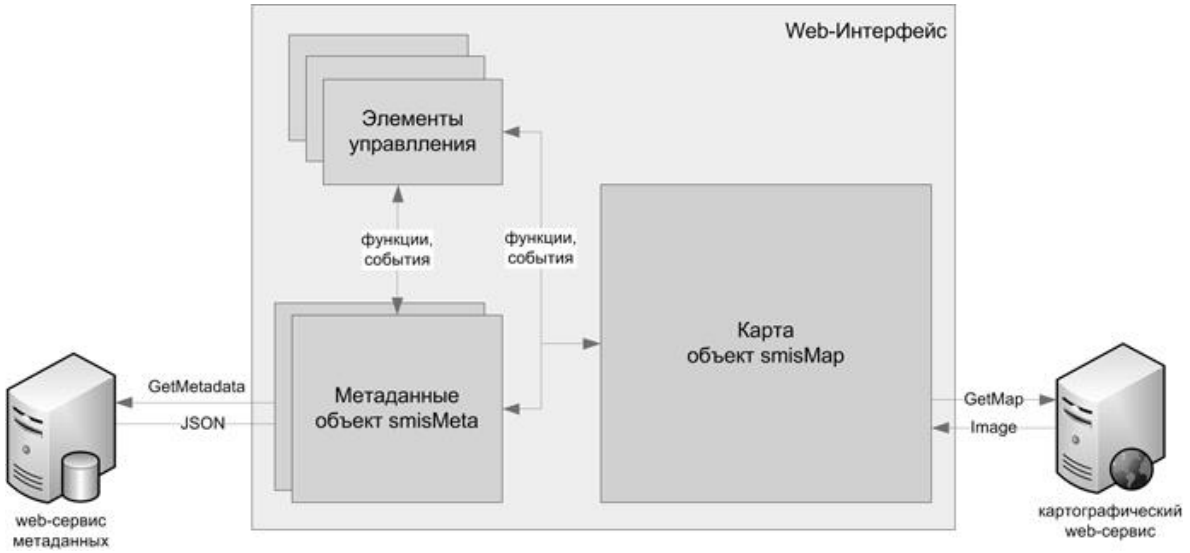


Рис. 3. Основные элементы архитектуры web-интерфейса

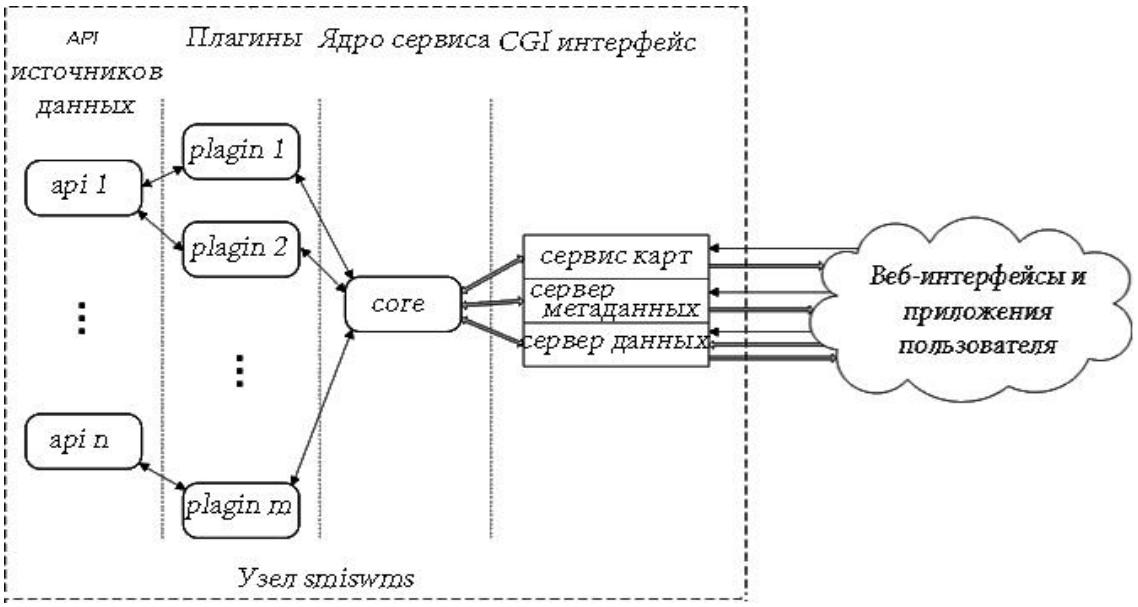


Рис. 4. Структурно-функциональная схема веб сервиса smiswms

Решение GETMAP (рис. 3), разработанное компанией «Совзонд», призвано максимально упростить процесс создания и внедрения собственной веб-ГИС за счет быстрого создания, не требующего навыков программирования или глубоких знаний ГИС; использования отечественной разработки и открытых программных продуктов, возможностей настройки карт, выполнения функций администраторов, операторов и пользователей системы; оперативной технической поддержки.

Другое направление решения задач хранения и анализа пространственных данных – тенденция переноса ресурсоемких приложений в среду облачных вычислений [13, 16], позволяющая использовать разработанные веб-ГИС в качестве интернет-сервисов, использующую распределенную трехзвенную архитектуру, позволяющую работать с пространственно-атрибутивными данными без необходимости их предварительной загрузки на веб-сервер благодаря сохранению в кэше веб-браузера программного кода, необходимого для обработки пространственно-атрибутивных данных ГИС на клиентской стороне приложения (ПО бизнес-логики веб-ГИС-клиента и ПО визуализации данных веб-ГИС-клиента) [6].

После инсталляции пользователи получают готовую веб-ГИС с уже подключенными базовыми картами, в которую можно загружать собственные данные, настраивать их отображение и использовать различный ГИС-инструментарий.

Компания «Совзонд» являющаяся официальным дистрибьютором ряда ведущих мировых компаний-операторов спутников ДЗЗ, ведущим российским интегратором в области геоинформационных технологий и аэрокосмического мониторинга была образована в 1992 году и активно занимается разработкой программных решений рассматриваемых задач [20].

Компания с 2007 года регулярно организует ГИС-форумы, проводит вебинары по обсуждению и ознакомлению пользователей со своими новыми продуктами, издает журнал «Геоматика». Она также предоставляет

пользователям данные и сервисы для добычи космической информации (см. http://catalog.sovzond.ru/, доступ к информационной системе, предназначенной для формирования структурированного архива пространственных данных «Геоаналитика.Архив» (рис. 5) – [20].

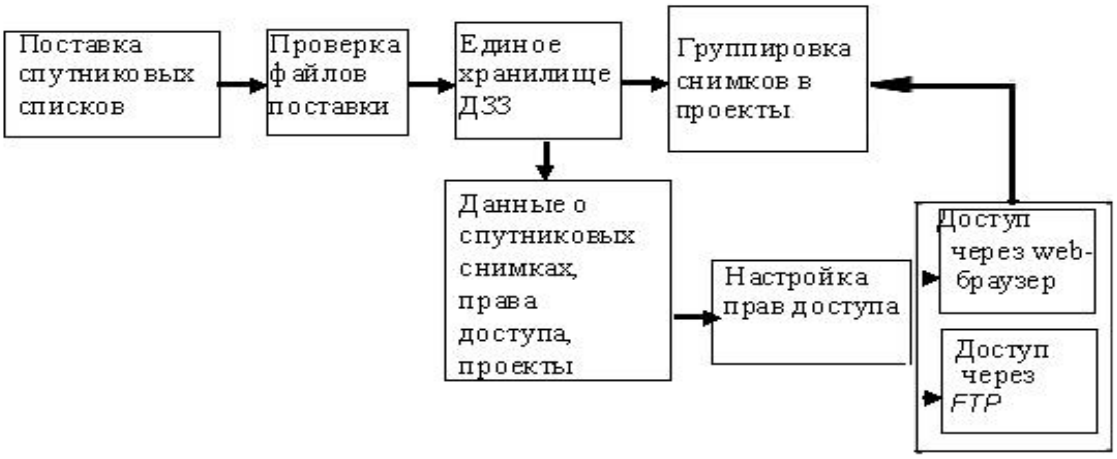


Рис. 5. Схема работы информационной системы Геоаналитика

Компания «Совзонд» является официальным дистрибьютором продуктов Airbus Defence and Space в России и странах СНГ, в том числе программы SpaceDataHighway для ускоренной передачи данных со спутников ДЗЗ, позволяющей благодаря лазерной технологии ежедневно передавать до 40 Тбайт данных со спутников ДЗЗ, авиационных систем или с Международной космической станции со скоростью 1,8 Гбит/сек. (Программа является результатом государственно-частного партнерства между Европейским космическим агентством и Airbus Defence and Space при спонсорстве Германского центра авиации и космонавтики.)

Предоставляемые программные продукты: фотограмметрическая обработка inpho, обработка данных с БПЛА, тематическая обработка ENVI, комплексная обработка TRSS, обработка радарных данных SARscape, гидрогеологическое моделирование Waterloo и др., услуги: проблемно-ориенти-рованное геомоделирование, создание ситуационных центров, комплексных информационно-аналитических систем, аппаратно-программных технологических комплексов, лабораторий ДЗЗ.

«Совзонд» планирует использовать в будущих разработках технологии Big Data и IoT (интернет вещей – «сеть сетей»).

Компания «СКАНЭКС» с 1989 года занималась разработкой станций приема спутниковых данных, затем создала более 30 программных продуктов, поддерживает on-line каталог космических изображений, в том числе оказывает услуги по предоставлению данных с КА Landsat 8 из глобального архива USGS, который содержит покрытие практически всей поверхности Земли, причем некоторые регионы, в том числе и вся территория России, отсняты многократно. Компания выполняет проекты мониторинга для Министерства природных ресурсов РФ, Министерства чрезвычайных ситуаций РФ, Росгидромета; десятки проектов в области сельского

и лесного хозяйства, нефтегазовой отрасли, мониторинга ЧС, экологического состояния территорий и акваторий, транспорта, строительства и недвижимости, кадастра и картографии, энергетики, регионального и муниципального управления и многих других. На базе технологий компании созданы центры космического мониторинга в ведущих вузах страны. Компания издает журнал «Земля из космоса». СКАНЭКС предоставляет доступ к КС с помощью интерфейса (http://www.scanex.ru/geo-service/onlayn-katalog/, kosmosnimki.ru.) [19].

Несмотря на некоторые успехи развития информационных технологий, поиск, выборка и анализ дистанционных данных до сих пор представляют собой недостаточно автоматизированный процесс последовательного просмотра имеющихся сайтов, предоставляющих доступ к изображениям по заданным характеристикам.

Ситуация в ближайшее время будет усложняться по мере расширения использования беспилотных летательных аппаратов для аэрофотосъемки (БПЛА) и необходимости сохранять получаемые данные для конкретных проектов в региональных базах [10]. Эти базы должны загружаться (желательно автоматически в режиме on-line), и к ним должен быть обеспечен доступ как для управления полетом, так и для выборки и анализа данных с учетом обеспечения конфиденциальности и защиты данных. Из накопленного опыта разработки каталогов и хранилищ спутниковых данных, очевидно, что, прежде всего, необходимо разработать и согласовать соответствующие стандарты протоколов, служебных функций и др. [11, 12]. Работу пользователей с данными в этом случае можно представить как на рис. 6.

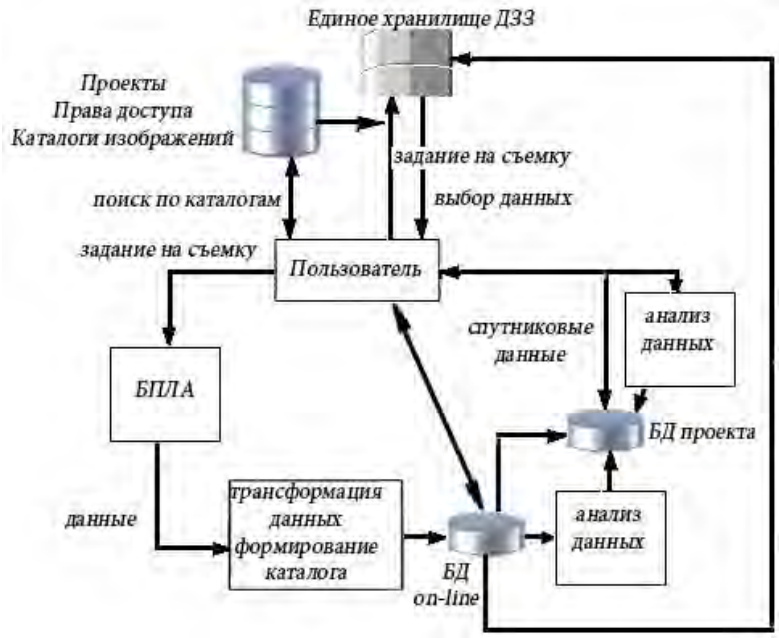


Рис. 6. Схема взаимодействия пользователя с хранилищами данных ДЗЗ

Выбор данных с помощью мобильных устройств также требует развития соответствующих стандартов и технологий. Например, удаленное выполнение программ можно реализовать с помощью предлагаемого в

настоящем сборнике подхода – использования утилиты WEBSOCKETD [9].

Заключение

Развитие аппаратуры и областей использования данных дистанционного зондирования Земли для решения множества хозяйственных и научных задач привело к резкому увеличению объема распределенной разномасштабной информации, получаемой в разных диапазонах электромагнитного спектра, предназначенной для «вечного хранения». Это требует развития новых технологий организации поиска, передачи и хранения получаемых данных, формирования каталогов уже накопленной информации. Используемыми в настоящее время подходами к решению задач является «просто» накопление и предоставление данных операторами и разработка проблемно-ориентированных интеллектуальных сервисов, в качестве перспективных рассматриваются технологии Big Data и IoT (компания «Совзонд»). В ближайшем будущем придется решать вопросы интеграции многочисленных распределенных данных БПЛА в «единые» хранилища данных о Земле.

Библиографический список

1. Антонов А.В., Бурцев М.А., Ефремов В.Ю., Калашников А.В., Крамарева Л.С., Крашенинникова Ю.С., Лупян Е.А., Матвеев А.М., Прошин А.А., Флитман Е.В**.** Построение объединенного каталога распределенных архивов спутниковых данных различных центров.// Современные проблемы дистанционного зондирования Земли из космоса, 2010. Т.7. № 2. С.84-89

2. Кудашев Е.Б., Филонов А.Н. Организация информационной распределенной среды и интеграция спутниковых архивов. Труды 7-ой Всероссийской научной конференции«Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL’2005, Ярославль, Россия, 2005. http://rcdl.ru/doc/2005/sek1_1_paper.pdf.

3. Кудашев Е.Б., Филонов А.Н. Технологии и стандарты интеграции сервисов, каталогов и баз данных дистанционного исследования Земли из космоса. Труды 9ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL’2007, Переславль-Залесский, Россия, 2007.

4. Кудашев Е.Б., Филонов А.Н. Формирование распределенной среды спутниковых исследований состояния природной среды. http://tm.ifmo.ru›tm2007/src/019d.pdf

5. Миллер С.А. Концепция российской инфраструктуры пространственных данных. http://www.agiks.ru/data/konf/page8.htm#top

6. Милихин М.М., Рычагов М.М. Веб-ГИС как интернет-сервис. http://gisconf.uriit.ru›materials/materials_gis2015.pdf

7. Толпин В.А, Балашов И.В., Ефремов В.Ю., Лупян Е.А., Прошин А.А., Уваров И.А., Флитман Е.В. Создание интерфейсов для работы с данными современных систем дистанционного мониторинга (система

GEOSMIS) // Современные проблемы дистанционного зондирования Земли из космоса. 2011. Т. 8. № 3. С. 93-108

8. Шокин Ю.И., Добрецов Н.Н,, Кихтенко В.А., Смирнов В.В., Чубаров Д.Л., Чубаров Л.Б. О распределенной инфраструктуре системы спутникового мониторинга ЦКП ДДЗ СО РАН. Вычислительные технологии, т.18, спецвыпуск, 2013, с. 86-94.

9. Хабаров С.П. Использование утилиты websocketd для удаленного выполнения программ. Настоящий сборник.

10. Шубина М.А. Использование беспилотных летательных аппаратов для аэрофотосъемки в целях картографирования наземных объектов. «Информационные системы и технологии: теория и практика». Сб. науч. тр. Выпуск 7. СПб.: СПбГЛТУ, 2015. с. 64-79.

11. Шубина М.А. Тенденции в разработке средств обработки аэрокосмических изображений (АКИ). «Информационные системы и технологии: теория и практика». Сб. науч. тр. Выпуск 4. СПб.: СПбГЛТУ, 2012. с.76-90.

12. Шубина М.А. Управление данными. Учебное пособие. СПб.: СПбГЛТУ, 2016. – 132 с.

13. Aly A.G., Labib N.M. Proposed Model of GIS-based Cloud Computing Architecture for Emergency System // International journal of computer science and mobile applications, 2014, i.1.

14. Hughes J. Steven and McMahon Susan K.. The Planetary Data System - Distributed Inventory System.
https://www.computer.org/csdl/proceedings/adl/1999/0219/00/02190086.pdf

15. http://en.academic.ru/dic.nsf/enwiki/107665

16. Steiniger S., Hunter A. J. S. Free and open source GIS software for building a spatial data infrastructure // Geospatial free and open source software in the 21st century. – Springer Berlin Heidelberg, 2012, p. 247-261.

17. http://www.infeo.org.

18. http://planet.iitp.ru

19. http://scanex.ru

20. http://sovzond.ru

А.Г. Шпекторов доцент
Фам Ван Туан, аспирант

ОБРАБОТКА НАВИГАЦИОННОЙ ИНФОРМАЦИИ В ЗАДАЧАХ УГЛОВОЙ СТАБИЛИЗАЦИИ МАЛОГАБАРИТНЫХ ПОДВОДНЫХ АППАРАТОВ

Введение

В настоящее время микромеханические измерительные системы находят широкое применение для задач навигации и управления. Особенно эти системы актуальны для подводных аппаратов, которые, ввиду малых

масс и габаритов, не предназначены для размещения высокоточных и громоздких навигационных средств измерения.

Объектом исследования в данном докладе является измерительный модуль MPU5060, в состав которого входит микромеханические гироскоп и акселерометр. Измерение проекций линейного ускорения по трем осям позволяет вычислять углы крена и дифферента модуля, гироскоп обеспечивает измерение соответствующих угловых скоростей. Однако микромеханические чувствительные элементы отличаются довольно низкой точностью и помехозащищенностью, поэтому для получения приемлемых оценок параметров движения необходимо использовать алгоритмы фильтрации.

Известный метод получения оценок углов и угловых скоростей основан на построении фильтра Калмана для модели двойного интегратора [1] и предполагает дальнейшее увеличение порядка модели для учета внешних возмущений. Однако, при наличии дополнительной информации о погрешностях измерений, математическая модель и фильтр Калмана могут быть получены в упрощенном виде.

Анализ измерительного модуля MPU6050.

Внешний вид измерительного модуля представлен на рис. 1. В состав модуля интегрирован трехосный акселерометр, трехосный гироскоп и цифровой процессор для первичной обработки измерений. В модуле MPU6050 предусмотрен аналоговый вход для внешнего компаса, что позволит обеспечить измерение ориентации трех углов Эйлера. Обмен данных поддерживается интерфейсной шиной I^2C. Встроенные в модуль аналогово-цифровые преобразователи (3 АЦП для акселерометра и 3 – для гироскопа) обеспечиввают измерения на выходе модуля в цифровом виде. Модуль программно может быть настроен на разные диапазоны измерения угловой скорости (±250, ±500, ±1000, и ±2000 градусов в секунду) и линейного ускорения (±2, ±4, ±8 и ±16 g). Погрешности измерений приведены в спецификации модуля [2].

Углы крена (θ) и дифферента (ψ) модуля могут быть получены по данным акселерометра:

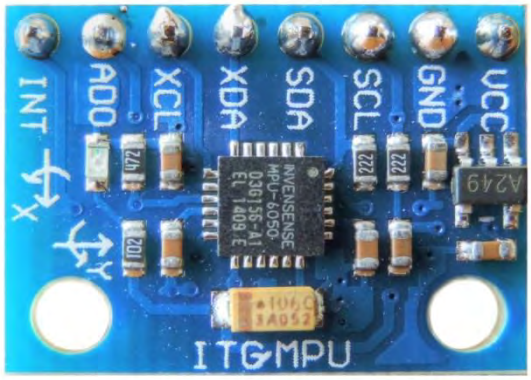


Рис 1. Структура MPU 6050

$$\theta = \text{arctg}\frac{a_y}{a_x};\ \theta = \text{arctg}\frac{a_z}{a_x}$$

где a_x, a_y, a_z – проекции ускорения на продольную, вертикальную и поперечную оси в системе координат, связанной с центром масс модуля.

Измерения акселерометра имеют существенные выбросы и искажения, обусловленные действием боковых ускорений.

Углы крена и дифферента также могут быть получены путем интегрирования соответствующих проекций угловой скорости, полученных от гироскопа, однако при этом возможно накопление ошибки интегрирования. Кроме того, для микромеханических гироскопов характерно мало меняющееся смещение угловой скорости. Соответственно, требуется провести фильтрацию измерений акселерометра и гироскопа, сочетающую достоинства обоих датчиков.

Математическая модель измерительного модуля и фильтра Калмана

Способы измерения угла крена и угла дифферента одинаковы, поэтому можно рассматривать модель измерения любого угла. Далее алгоритм фильтрации будет рассматриваться для измерений угла крена. Поскольку измерение угловой скорости гироскопом на малом интервале времени имеет довольно высокую точность, его можно использовать для коррекции измерения акселерометром. Математическая модель угла крена модуля имеет вид:

$$\frac{d\theta(t)}{dt} = \overline{\omega}_x(t) - \omega_{xb}(t), \tag{1}$$

где $\overline{\omega}_x$ – измерение угловой скорости гироскопом; ω_{xb} – угловая скорость дрейфа гироскопа.

Поскольку угловая скорость ухода гироскопа меняется мало, ее можно описать уравнением

$$\frac{d\omega_{xb}(t)}{dt} = \mu(t), \tag{2}$$

где $\mu(t)$ – гауссовый белый шум с малой ковариацией.

Модуль MPU6050 обеспечивает измерения в цифровом виде, поэтому уравнения (1) и (2) удобнее представить в дискретной форме:

$$\begin{bmatrix} \theta \\ \omega_{xb} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & -h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \omega_{xb} \end{bmatrix}_k + \begin{bmatrix} h \\ 0 \end{bmatrix} \overline{\omega}_{x_k}, \tag{3}$$

где h – период дискретизации.

Уравнение выхода имеет вид:

$$\bar{\theta}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega_{xb} \end{bmatrix}_k + \nu_k, \tag{4}$$

где ν_k – гауссовый белый шум, описывающий погрешность измерений угла крена. Ковариационные матрицы процессов, описывающих погрешность измерений угловой скорости (Q_ω) и угла крена (R_θ), могут быть заданы, исходя из характеристик модуля.

Рассмотрим построение дискретного фильтра Калмана для модели (3), (4), работающего по принципу прогноза-коррекции [3]. Априорная оценка вектора состояния модели (3), а также матрица ковариации ошибки имеет вид

$$\hat{x}_{k+1}^{-} = A_d \hat{x}_k^{-} + B_d u_k,$$

$$P_{k+1}^{-} = A_d P_k A_d{}^T + Q_k, \tag{5}$$

где A_d, B_d – дискретные матрицы состояния и управления (3); $\hat{x}_k^{-}$ – априорная оценка вектора состояния (3); $u_k = \bar{\omega}_{x_k}$; P_k – матрица ковариации ошибки оценивания; Q_k – матрица ковариации, определяемая погрешностью измерения угловой скорости:

$$Q_k = Q_\omega^2 B_d B_d{}^T = \begin{bmatrix} (Q_\omega h)^2 & 0 \\ 0 & 0 \end{bmatrix}.$$

Скорректированная оценка фильтра Калмана имеет вид:

$$\hat{x}_{k+1} = \hat{x}_{k+1}^{-} + K_{k+1}\left(y_{k+1} - C_d \hat{x}_{k+1}^{-}\right);$$

$$K_{k+1} = P_{k+1}^{-} C_d{}^T \left(C_d P_{k+1}^{-} C_d{}^T + R\right)^{-1}, \tag{6}$$

где C_d – матрица наблюдения уравнения (4); K_k – матрица коэффициентов фильтра Калмана; $y_k = \bar{\theta}_k$; $R = R_\theta$.

Ковариация ошибки фильтра с коррекцией может быть определена по формуле:

$$P_{k+1} = \left(I - K_{k+1} C_d\right) P_{k+1}^{-}, \tag{7}$$

где I – единичная матрица.

Выражения для матриц состояния, управления и наблюдения математической модели имеют простую форму, поэтому выражения (5)–(7) можно определить аналитически.

$$P_{11}^{-} = P_{11} + h\left(P_{22}h - P_{12} - P_{21} + {Q_\theta}^2 h\right);$$
$$P_{12}^{-} = P_{12} - P_{22}h;$$
$$P_{21}^{-} = P_{21} - P_{22}h;$$
$$P_{22}^{-} = P_{22};$$
$$K_1 = \frac{P_{11}^{-}}{1 + P_{11}^{-}};$$
$$K_2 = \frac{P_{21}^{-}}{1 + P_{11}^{-}};$$
$$\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}_{k+1} = \begin{bmatrix} P_{11}^{-}(1-K_1) & P_{12}^{-}(1-K_1) \\ P_{21}^{-} - K_2 P_{11}^{-} & P_{22}^{-} - K_2 P_{21}^{-} \end{bmatrix},$$

где $P_{11}, P_{12}, P_{21}, P_{22}$ – элементы матрицы P_k; K_1, K_2 – элементы вектора коэффициентов K_{k+1}. В качестве начальных значений матрицы P_k можно выбрать нулевые значения. Оценка угла крена, восстановленная фильтром Калмана, вычисляется следующим образом:

$$\theta_{k+1}^{-} = \theta_k^{-} + h\left(\overline{\omega}_{x_k} - \omega_{xb_k}\right);$$
$$\theta_{k+1} = \theta_{k+1}^{-} + K_1\left(\overline{\theta}_{k+1} - \theta_{k+1}^{-}\right);$$
$$\omega_{xb_{k+1}} = \omega_{xb_k} + K_2\left(\overline{\theta}_{k+1} - \theta_{k+1}^{-}\right).$$

Упрощенные формулы для расчета коэффициентов фильтра Калмана, матрицы ковариаций и оценок можно легко реализовать с помощью стандартных средств высокоуровневых языков программирования. Фильтр аналогичной структуры может быть реализован независимо для получения оценки угла дифферента, а при наличии внешнего компаса – также оценки угла рыскания.

Результаты испытаний модуля MPU6050 с фильтром Калмана

В качестве исходных параметров для фильтра Калмана, согласно настройкам модуля MPU6050 были выбраны следующие ковариации:

$Q_\omega = 0.04$; $R_\theta = 0.05$.

Результаты испытаний измерительного модуля без обработки фильтром приведены на рис. 2 (для неподвижного состояния модуля). Из рис. 2 видно, что отдельные выбросы измерения акселерометром достигают 1.5°, а гироскоп имеет дрейф (угловая скорость составляет примерно 0.005 °/с).

На рис. 3 приведены результаты работы фильтра Калмана при повороте модуля вдоль оси *x*. Из рисунка видно, что фильтр обеспечивает сглаженную оценку угла, с точностью до 0.2°. На рис. 4 и рис. 5 приведены результаты измерений параметров модуля при переменном воздействии

Из рисунков видно, что фильтр Калмана обеспечивает достаточно стабильную точность восстановления угловой ориентации модуля. Таким образом, модуль MPU6050 с алгоритмами калмановской фильтрации может быть использован для измерения угловых параметров движения и решения задач стабилизации крена и дифферента.

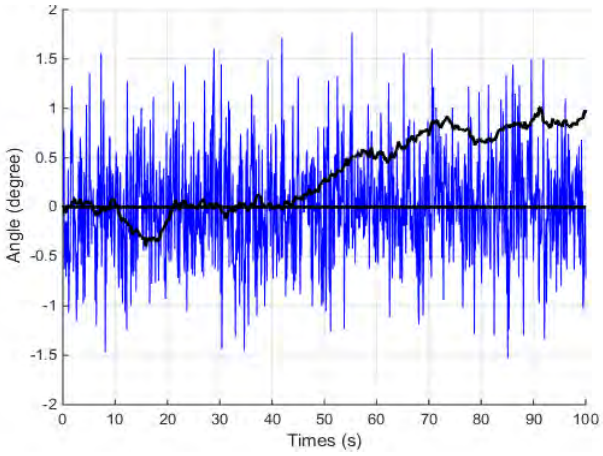


Рис 2. Данные акселерометра и гироскопа без фильтра

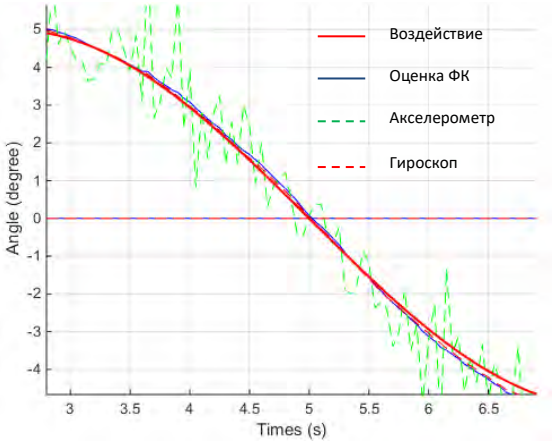


Рис 3. Оценка поворота модуля с фильтром

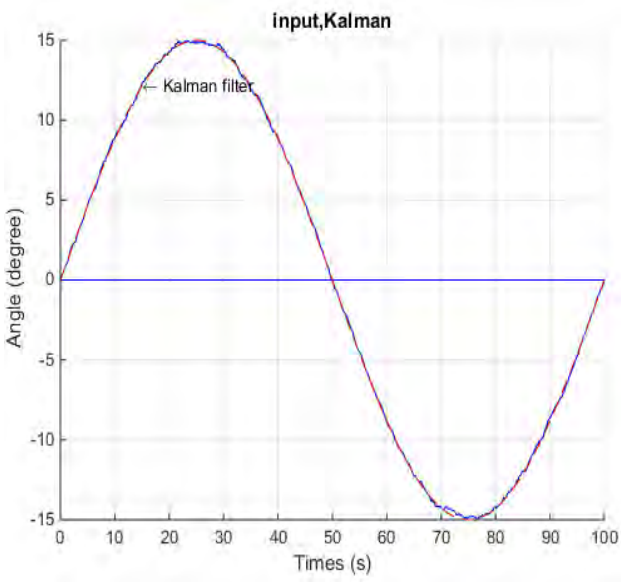


Рис 4. Оценка переменного воздействия фильтром Калмана

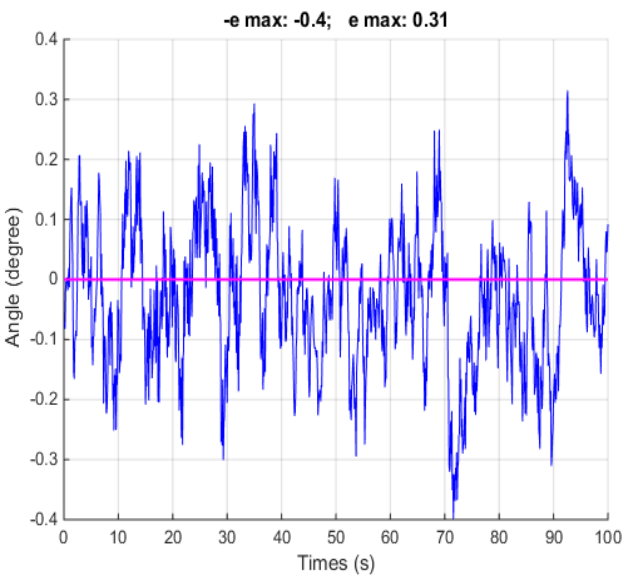


Рис 5. Динамическая ошибка фильтра Калмана

Библиографический список

1. Хабаров С.П., Коренев А.С., Амбросовский В.М. Отказоустойчивый адаптивный к внешним возмущениям фильтр Калмана // Морская радиоэлектроника-СПб .:Отраслевые журналы ,2015. № 3, с. 20-23

2. MPU-6000 and MPU-6050 Product Specification. url: www.cdiweb.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf

3. Brandon McCarron. Low-Cost IMU Implementation via Sensor Fusion Algorithms in the Arduino Environment / url: http://digitalcommons.calpoly.edu /cgi/viewcontent.cgi?article=1114&context=aerosp.

А.Х. Шоюн, студентка 3 курса
И.В. Ганичев, кандидат.технических наук, доцент
С.В. Киселева, доцент

ОРГАНИЗАЦИЯ ПРАКТИКУМА В КУРСЕ «АРХИТЕКТУРА ЭВМ И ИНФОРМАЦИОННЫХ СИСТЕМ»

Цель данной работы – интенсификация интерактивных занятий и самостоятельности студентов. Выбрана технология практического обучения с помощью компьютерного симулятора ЭВМ.

Методика основана на исследованиях и системном анализе структур ЭВМ, вовлечении в информационный процесс общепризнанной, наглядной и лучшей, по мнению экспертов, на данное время учебной модели ЭВМ [1].

Расчетно-аналитические работы выполняются по дисциплине в среде Mathcad. Общее действие методик сводит воедино понимание функциональных задач, связи программирования и компиляции с микропрограммированием в структуре команд и композиции ряда аппаратурных решений. Дисциплина базируется на знаниях, приобретенных в курсах информатики, компьютерных сетей, теории информационных процессов и систем, а также моделирования информационных систем.

Знание функциональной и структурной организации компьютера является целевой основой формирования профессиональных компетенций в процессе обучения и далее для любой сферы деятельности выпускника высшей школы.

Перед практическим занятием приводятся проблемные вопросы и постановка задач в формате мини-лекций, а на практическом занятии постоянно практикуется популярная стратегия работы в малых группах по требованию и необходимости разрешения возникающих разногласий. Это учит самостоятельности в мышлении. В диалогах мы активизируем вопросы и обсуждаем основные принципы создания, достоинства и недостатки тех или иных структурных решений.

Представление о системной организации ЭВМ формируются как концепция архитектурной модели в описаниях возможностей и композиций составных частей, использования технических компонентов. При этом в нашем понимании раскрываются особенности внутреннего конвейера команд, технологических подходов и конструктивных решений параллельных вычислений и достаточно сложных структур.

По нашему мнению, в классификации структур следует использовать кроки *схем,* расположенных *по квадрантам,* с упрощенным набором классифицирующих признаков – по одиночным потокам команд *SI* (ОК) и множественным *MI* (МК), затем одиночным потокам данных *SD* (ОД) и множественным *MD* (МД) – в четырех сочетаниях М. Флинна, что зримо оттеняет идею параллелизма. Наименования векторной, конвейерной и матричной структуры многопроцессорных ЭВМ хорошо воспринимаются и легко анализируются в данном комплексе.

Архитектурные решения рассматриваются на различных уровнях описания, со свои компонентами (и разными категориями языков описания) в иерархическом представлении страт, выделенных исследователями [4].

В ходе лабораторного практикума у студентов закрепляются базовые понятия: компьютерной архитектуры, изучаемой классификации компьютерных систем (рис. 1), на разных уровнях архитектуры, принципов действия и взаимодействия подсистем.

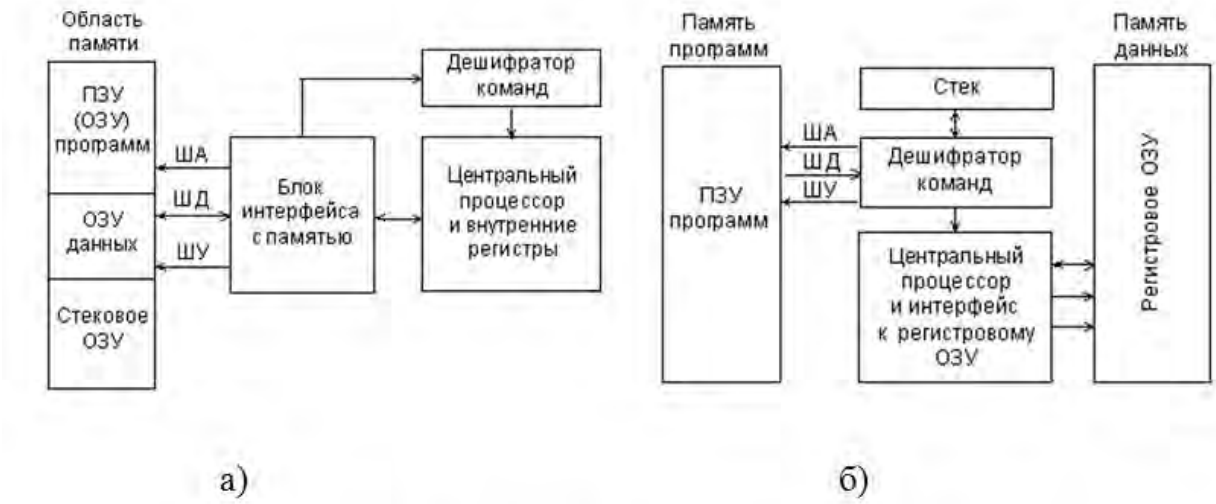


Рис1. Классическая архитектура а) фон Неймана, б) гарвардская
http://www.intuit.ru/studies/courses/3/

Учебная модель реализует классическую неймановскую архитектуру на базе эмуляции сокращенного набора машинных команд RISC. В отличие от упрощенного рассмотрения реально существующих их прототипов применяется исходная простая модель – симулятор, что облегчает понимание принципов взаимодействия компонентов компьютера на уровне функциональных устройств, а также на уровне архитектуры машинных команд и ассемблера, это важно и для системных программистов.

Рассматриваются система команд процессора, типы и форматы команд/операндов, способы адресации операндов, наборы регистров, взаимодействие с устройствами ввода/вывода, обработка особых ситуаций.

В лабораторном практикуме упор делается на применении программ, инициирующих работу симуляторов, что повышает наглядность закрепля-

емого учебного материала и облегчает восприятие учебного материала студентами.

Работа состоит в освоении архитектуры и системы команд, их форматов, описания модели арифметико-логического устройства.

Взаимодействие рассматривается на следующих уровнях: операционных блоков и управляющих автоматов, а также базовых подсистем: процессоры – запоминающие устройства, системные магистрали, устройства ввода-вывода.

Исследуется командный цикл процессора в виде последовательности микрокоманд, которая выводится в специальном окне «микрокомандный уровень». Исследователь легко вычисляет характеристики модели внутреннего конвейера.

Работа выполняется поэтапно. Конструируется система из основных блоков. Проверяются на тестах устройства во взаимосвязях:
а) процессор – основная память (ОЗУ), контроллер обмена; б) кэш память (СОЗУ), вначале она отключена. Подключаются внешние устройства: а) контроллер – дисплей, б) обозреватель информации.

Уникально исследование эффективности использования кэш-памяти в общей системе памяти. Она добавляется в исследуемую композицию по команде интерфейса симулятора «Кэш включена». На основе предлагаемого эксперимента изучаются в учебном процессе гипотезы влияния ряда факторов: емкости кэш-памяти; размера строки; способа отражения ОП на кэш-память; алгоритма замещения информации в заполненной кэш-памяти и алгоритма согласования содержимого памяти [2,3].

Целью данного исследования будет проверка работы различных алгоритмов замещения при различных режимах записи. Фиксируются признак занятости ячейки тега, признак использования и признак записи в ячейку по алгоритмам сквозной записи или обратной в сочетании с назначенными алгоритмами замещения строк. В модели реализованы три различных алгоритма замещения строк:

случайное, когда номер ячейки кэш-памяти выбирается случайным образом;

очередь, при которой выбор замещаемой ячейки определяется временем пребывания ее в кэш-памяти;

бит использования, когда случайный выбор осуществляется только из тех ячеек, которые имеют нулевое значение флага использования.

Набирается статистика кэш-попаданий и соответствующим определяется коэффициент K эффективности работы кэш-памяти.

Заключение. В отличие от упрощенного рассмотрения реально существующих прототипов ЭВМ применение технологии практического обучения с помощью компьютерного симулятора ЭВМ облегчает понимание принципов взаимодействия компонентов компьютера, создает переход к системной картине эволюции мира современных ЭВМ [5]. Студенты приобретают навыки по выбору, настройке и использованию современных компьютеров для решения прикладных задач.

Библиографический список

1. Учебные модели компьютера [Электронный ресурс]. URL: http://educomp.runnet.ru/model.

2. Архитектура ЭВМ и информационных систем. Структурная организация / И.В. Панфилов, А.М. Заяц. – СПб.: СПбГЛТУ, 2013. – 96 с.

3. Архитектура ЭВМ и информационных систем. Функциональная организация / И.В. Панфилов, А.М. Заяц. – СПб.: СПбГЛТУ, 2013. – 96 с.

4. Ганичев И.В., Троицкая Е.Г., Троицкая А.Г. Страты ЭВМ. /Системный анализ в проектировании и управлении. Сборник научных трудов XIII Международ-ной научно-практической конференции. СПбГПТУ, 2009.

5. Хабаров С.П., Шилкина М.Л. Вычислительные машины, системы и сети. – СПб.: СПбГЛПТУ, 2017. – 240 с.

6. Жмакин А.П. Архитектура ЭВМ: 2-е изд., перераб. и доп.: учеб. пособие. – СПб.: БХВ-Петербург. 2010. – 352 с.

ОГЛАВЛЕНИЕ

Научное издание

Отв. редактор
Заяц Анатолий Миосеевич

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ: ТЕОРИЯ И ПРАКТИКА

С б о р н и к н а у ч н ы х т р у д о в

Выпуск 9

Отпечатано в авторской редакции с готового оригинал-макета

Подписано в печать с оригинал-макета
Форомат 60x84/16. Бумага офсетная. Печать трафаретная.
Уч.-изд.л. 5,0. Печ. л. 5,0. Тираж 100 экз. Заказ №

Санкт-Петербургский государственный лесотехнический университет
Издательско-полиграфическкский отдел СПбГЛТУ
194021, Санкт-Перербург, Институтский пер., 3