МИНОБРНАУКИ РОССИИ
федеральное государственное учреждение
образовательное учреждение
высшего образования
«ЮГО-ЗАПАДНЫЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ» (ЮЗГУ)

Сборник материалов
II Всероссийской научно-практической конференции
**«ПРОГРАММНАЯ ИНЖЕНЕРИЯ:
СОВРЕМЕННЫЕ ТЕНДЕНЦИИ
РАЗВИТИЯ И ПРИМЕНЕНИЯ»**

11-12 марта 2018 года

Ответственный редактор - *Томакова Р.А.*

КУРСК - 2018

УДК 004
ББК 32.97

Редакционная коллегия:
Томакова Р.А.

Программная инженерия: современные тенденции развития и применения: Сборник материалов Всероссийской конференции (11-12 марта 2018 г.). – Курск: ЗАО «Университетская книга», 2018 г. - 127 с.

ISBN 978-5-907049-37-6

В сборнике представлены материалы докладов Всероссийской конференции, прошедшей в Юго-Западном государственном университете на кафедре «Программная инженерия» 11-12 марта 2018 г. Доклады охватывают широкий спектр проблем в области создания, проектирования, анализа, моделирования и оценки информационных систем различного назначения, а также ряд вопросов, касающихся разработки и внедрения новых информационных технологий.

ISBN 978-5-907049-37-6

УДК 004
ББК 32.97

ОГЛАВЛЕНИЕ

Аболмасова Е.С., студент, e-mail: evgeniya.abolmasova@gmail.com
Емельянова Е.С., студент, e-mail: eemelyanova12@yandex.ru
Шепелева Ж.А., студент, e-mail: schepelewa.janna@yandex.ru
ЮЗГУ, г. Курск, Российская Федерация

ПРЕИМУЩЕСТВА И ПРИНЦИПЫ РАЗРАБОТКИ В CMSWORDPRESS

В статье рассмотрены основные преимуществаи принципы работы в CMSWordPress, которая предназначена для управления содержимым сайтов с открытым исходным кодом. Отмечены достоинства и недостатки использования данной платформы.

***Ключевые слова**: WordPress, Content Management System (CMS), сайт, плагин, платформа.*

THE ADVANTAGES AND PRINCIPLES OF DEVELOPMENT IN CMS WORDPRESS

The article describes the main advantages and principles of working in CMS WordPress, which is designed to manage the content of open-source sites. Advantages and disadvantages of using this platform are noted.

***Key words**: WordPress, Content Management System, website, plugin, platform.*

В настоящее время широкое распространение получила система управления содержанием сайта с открытым исходным кодом – Wordpress. Возможности, которые предоставляет движок WordPress действительно огромные, проект динамично развивается, появляются новые версии, плагины и сервисы, делающие движок WordPress оптимальным решением, как для ведения бизнеса, так и для самовыражения (например, для ведения блога). На рисунке 1 проиллюстрирован рейтинг среди наиболее известных CMS.

История появления самой популярной системы для ведения автономных блогов WordPress ведет свое начало с 2001 года. Именно тогда М. Валдриги приступил к разработке движка b2, который является основой CMS WordPress. Затем к разработке проекта b2 присоединились другие исследователи, однако создателем WordPress принято считать М. Муленвега,

который ввёл новое название проекта - WordPress и получил права на торговую марку [1-2].

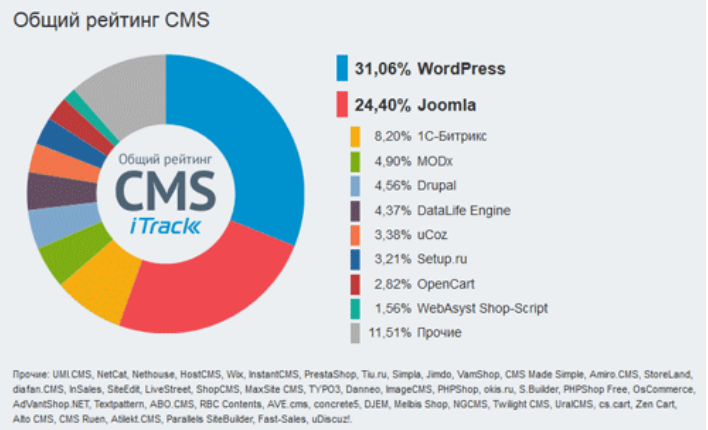


Рис.1 – Общий рейтинг CMS

В 2003 году состоялся первый релиз WordPress, в дальнейшем практически ежегодно выходили улучшенные версии WordPress, которые делали продукт все более удобным для пользователя, за счет того, что: добавлено правописание и автосохранение; улучшен интерфейс; появилась поддержка виджетов и тэгов; функция отслеживания изменений и управление темами из панели администратора; встроенграфический редактор и пакетное обновление плагинов.

WordPress реализован на языке PHP, использует системы MySQL и CSS, представляет собой конструктор, быстро наполняемый новыми сложными функциональными возможностями, на котором можно разрабатывать абсолютно любой по сложности и наполнению сайт [3-9].

Wordpress поддерживает технологии RSS, Ping, Trackback, ATOM и ЧПУ, расширяется с помощью плагинов, для нее разработано большое количество шаблонов и виджетов. Следует заметить, что архитектура CMS позволяет создавать визитки, блоги, магазины, галереи, социальные сети, портфолио и другие проекты. Большинство разработанных шаблонов и расширений для Wordpress, в том числе, бесплатно распространяемых, представлены в репозитории, поиск по которому осуществляется, используя возможности из панели администратора.

В связи с этим следует отметить основные достоинстваWordpress:

1) контент сайта доступен при использовании обычного браузера.

2) основным редактором wordpress является модель WYSIWYG (от анг.What You See Is What You Get). При создании и изменении страницы web-мастер сразу видит конечный результат.

3) меню и страницы меняются автоматически при добавлении новых материалов.

4) разработчики CMS предоставляют квалифицированную техническую и информационную поддержку.

5) для wordpress написано более 7 тысяч различных плагинов. Модуль достаточно закачать на сайт через FTP и активировать при помощи web-интерфейса.

6) в ходе продвижения сайта его дизайн можно изменять любое количество раз, не оказывая влияния на CSM.

7) создаваемые в Wordpress страницы соответствуют стандартам W3C для CSS и XHTML, как результат отсутствие необходимости в HTML валидаторах.

8) посетители сайта, созданного на Wordpress, могут оставлять комментарии к статьям, что улучшает интерактивность ресурса (взаимодействие авторов и целевой аудитории).

9) сайт на Wordpress может иметь любое количество страниц, что упрощает работы по его поисковой оптимизации. Каждой записи может быть присвоена определенная рубрика, для чего предусмотрена отдельная закладка. Категории могут иметь различную древовидную иерархию. Для дополнительной тематической группировки используются метки.

10) для безопасности сайта и снижения нагрузки на сервер для Wordpress написан антивирусный плагин, предусмотрено кэширование данных. Различные уязвимости устраняются разработчиками достаточно оперативно.

Следует также подчеркнуть важное преимущество использования системы: сайты на Wordpress в достаточной мере защищены от заражений, благодаря разработчикам и простой установке официальных обновлений системы.[3]

Вместе с этим, необходимо отметить недостатки, присущие Wordpress, такие как:

1) достаточно весомая нагрузка на сервер;

2) порою низкая скорость загрузки сайта;

3) ограниченный набор базовых функций;

4) дублирование страниц и изображений;

5) большое количество некачественных тем и дополнений с ошибками;

6) относительное недоверие поисковых систем к сайтам на бесплатных движках.

Принимая во внимание тот факт, что движок Wordpress, а особенно большое количество дополнений к нему могут существенно нагружать сервер, на котором располагается сайт. В связи с этим, для устранения возможности чрезмерной нагрузки, необходимо сокращать количество размещаемых плагинов до нужного минимума. Также одним из недостатков WordPress является то, что он автоматически генерирует дублирование страниц, а это негативно сказывается на продвижении сайта. Также движок Wordpress при загрузке изображений создает их копии разных размеров.

Вывод: широкие возможности, которые предоставляет движок WordPress, по достоинству оценили профессионалы индустрии. В 2007 году платформа WordPress победила на PacktOpenSource CMS Award. В 2009 году была признана лучшей на Open Source CMS Award. 2010 год - победа в категории «Зал славы CMS» на OpenSourceAwards. В 2011 WordPress был признан открытым веб-приложением года на ежегодном конкурсе The Critters.

Список использованных источников

1. Сабин-Вильсон, Л. WordPress для чайников / Л. Сабин-Вильсон. – Вильямс.2010. – 368 с.

2. Хассей, Т. UsingWordPress / Т. Хассей. – М.: Эксмо.2012. – 432 с.

3. Грачев, А. Создаем свой сайт на WordPress/ А. Грачев. Спб.: Питер.2013. – 272 с.

4. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Запюгосюун-т. – Курск, 2018. –258с.

5. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

6. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

7. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/ А.В. Малышев// Известия Юго-Западного государственного университе-

та.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

8. Белобров, А.П. Нейросетевые модели морфологических операторов для сегментации изображений медицинских сигналов/ А.П. Белобров, С.А. Борисовский, Р.А. Томакова//Известия ЮФУ. Технические науки. 2010.№8(109). –С.28-32.

9. Томакова, Р.А. Метод обработки сложноструктурируемых изображений на основе встроенных функций среды MATLAB/Р.А. Томакова, С.А. Филист//Вестник Забайкальского государственного университета. 2012. №1. –С.3-9.

Алексеев В.А., студент, e-mail: vladislaw.al2015@yandex.ru,
Макашин В.А., студент, e-mail: v.makashin@yandex.ru
ЮЗГУ, г.Курск, Российская Федерация

СОВЕРШЕНСТВОВАНИЕ РАСЧЕТНЫХ РАБОТ В ОБЛАСТИ ПРОЕКТИРОВАНИЯ ГРУЗОЗАХВАТНЫХ УСТРОЙСТВ И ПРИСПОСОБЛЕНИЙ

В статье кратко приведено описание программы расчёта такелажных скоб, используемых в операциях по подъему и перемещению грузов как соединительных элементов отдельных звеньев различных грузозахватных устройств.

Ключевые слова: *такелажная скоба; расчет такелажной скобы; программа расчета*

IMPROVING DESIGN WORK IN THE DESIGN OF LIFTING DEVICES AND FIXTURES

The article briefly describes the program of calculation of rigging brackets used in operations for lifting and moving loads as connecting elements of individual units of various load gripping devices.

Keywords: lifting clamp; lifting brackets calculation; the calculation program

Современные строительные площадки представляют собой высокомеханизированное производство, в ходе которого применяются десятки видов специализированной строительной техники. Использование грузоподъемных машин и технологической оснастки регламентируется различными предписаниями. Однако на практике часто эксплуатируется технологическая оснастка, находящаяся в неисправном или сильно изношенном состоянии, например, траверсы, стропы, которые требуют постоянного ремонта или замены. Высокий уровень производственного травматизма в строительном комплексе [4, 5] свидетельствует о большом количестве грубых нарушений законодательных, нормативно-правовых актов по охране труда и правил безопасности [6, 7]. Этому способствует низкая организационная и проектная культура в организациях строительства [2, 8].

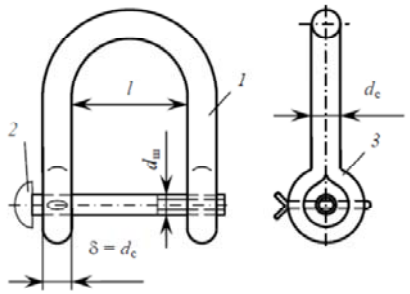
Актуальность работы обусловлена необходимостью снижения производственного травматизма и аварийности при эксплуатации грузоподъемных кранов на объектах строительства [3, 10].

Целью работы являются разработка программы для проверки параметров такелажных скоб при их конструировании или выборе для грузозахватных приспособлений.

Погрузочно-разгрузочные, транспортировочные, монтажные и складские работы весьма трудоемки и занимают значительный объем в строительстве и являются травмоопасными. Основные причины травматизма при монтажных работах - падение людей с высоты, падение монтируемых конструкций и перемещаемых грузов, падение приспособлений и инструмента на людей (достигает 70-75% от случаев травматизма при выполнении этого вида работ).

Травматизм на разгрузочных и складских работах составляет 12−17% общего числа несчастных случаев в строительстве. Причем значительная доля из них приходится на период производства монтажных работ. В 27% случаев они происходят по причине неисправности грузозахватных устройств, приспособлений и, в частности, из-за разрушения такелажных скоб, пальцев, осей шарниров и проушин. Падение груза происходит из-за износа или по причине выбора ненадлежащего типоразмера названных элементов грузозахватных приспособлений.

При подъеме и перемещении грузов такелажные скобы и другие перечисленные приспособления применяют в качестве соединительных элементов отдельных звеньев различных грузозахватных устройств, а также как самостоятельные захватные приспособления (рис. 1).



1 - ветвь скобы; 2 - штырь; 3 - бобышка

Рисунок 1 – Скоба такелажная

Технически грамотное перемещение грузов при условии обеспечения безопасности ведения монтажных работ связано с расчѐтом оснастки. Расчѐт такелажных средств и оснастки сводится к решению следующих двух задач.

1. Определение максимальных расчѐтных усилий, возникающих в различных элементах такелажных средств в процессе подъѐма и перемещения грузов.

2. Определение конструктивных размеров элементов с учѐтом максимальных нагрузок, действующих на них, или подбор стандартных элементов по расчѐтным нагрузкам.

С целью автоматизации расчетных работ разработана соответствующая программа, которая, позволяет, исходя из нагрузки, действующей на скобу, задавшись размерами элементов, проверить еѐ на прочность. Этот расчѐт выполняется в следующем порядке.

1. Зная нагрузку, действующую на скобу, например, от веса поднимаемого груза или усилия в грузовом канате, задав размеры элементов, условия работы находят усилие, действующее на скобу.

$P = Sk_nk_д$,

где S – нагрузка, действующая на скобу (например, вес поднимаемого груза, усилие натяжения каната и т.п.); k_n –коэффициент перегрузки; $k_д$ – коэффициент динамичности нагрузки. В данном случае расчет выполнен для нагрузки S = 140 кН.

2. Проверяют на прочность ветви скобы выбранного типоразмера.

3. Определяют изгибающий момент в штыре.

4. Находят момент сопротивления сечения штыря.

5. Проверяют штырь скобы на прочность при изгибе.

6. Проверяют штырь скобы на срез.

7. Проверяют отверстие скобы на смятие.

Расчетные формулы и справочные значения переменных параметров, входящих в расчетные формулы, приняты из работы [1] и в данной статье не приводятся.

На рис. 2 показан общий вид интерфейса программы для расчета параметров такелажных скоб.

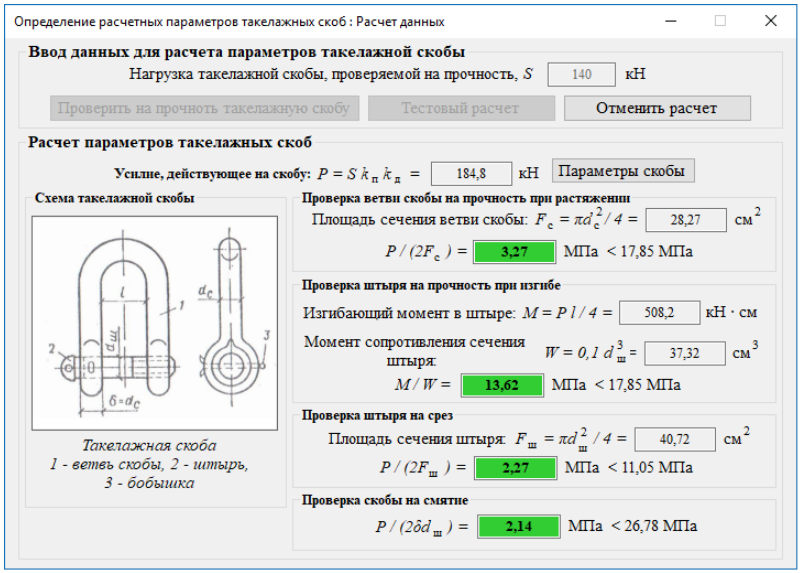


Рисунок 2 − Общий вид интерфейса программы

Вывод. Совершенствование расчетных работ, инновации в области проектирования и применение более совершенных грузозахватных устройств позволит снизить травматизм и расходы, связанные с несчастными случаями, а также будет способствовать росту производительности труда в строительстве [9].

Список использованных источников

1. Инженерные решения по охране труда в строительстве / Г.Г. Орлов, В.И. Булыгин, Д.В. Виноградов и др.; Под ред. Г.Г. Орлова. М.: Стройиздат, 1985 . 278 с.

2. Малые предприятия строительства и большие проблемы безопасности труда / В.И. Томаков, С.И. Меркулов // Проблемы обеспечения безопасности строительного фонда России: материалы IV Международных академических чтений. Курск, 2005. С. 251–261.

3. Состояние промышленной безопасности при эксплуатации грузоподъемных кранов на объектах, подконтрольных Ростехнадзору // Известия Юго-Западного государственного университета. Серия: Техника и технологии. 2017. Т. 7. №1 (22). С. 27-41.

4. Состояние условий труда, профессиональные заболевания и производственный травматизм в экономике Российской федерации / И.А. Томакова, В.И. Томаков // Известия Юго-Западного государственного университета. Серия: Техника и технологии – 2016. № 2 (19). С. 95–107.

5. Томаков В.И. Производственный травматизм в строительной отрасли // Безопасность жизнедеятельности. 2006. № 3. С.13–22.

6. Томаков, В. И., Томаков, М. В., Зубков, М. Э. Организационная культура охраны труда в строительной отрасли. Курск, 2012. 120 с.

7. Томаков, М. В. Нормативно-правовая основа системы управления охраной труда и промышленной безопасностью организаций строительства // Известия Юго-Западного государственного университета. Серия «Техника и технологии». 2012. №2-3. С. 248−252.

8. Томаков, М.В., Томаков, В.И., Казакова, Ю.М. Тенденции, причины производственного травматизма и мероприятия по улучшению охраны труда и управления персоналом организаций [Текст] // Актуальные проблемы экологии и охраны труда. Сб. статей VII Международной научно-практической конференции. Юго-Западный государственный университет. Курск, 2015. С. 314–319.

9. Экономика и социология труда / И. А. Томакова, П. И. Почечун, М. В. Томаков. Курск, 2016. 137 с.

10. Tomakov V.I., Tomakov M.V., Pahomova E.G. and other. A study on the causes and consequences of accidents with cranes for lifting and moving loads in industrial plants and construction sites of the Russian Federation // Journal of Applied Engineering Science. 16 (2018) 1, 504. P. 95-98.

Алексеев В.А., студент, e-mail: vladislaw.al2015@yandex.ru,
Алябьев С.А., студент, e-mail: dold4712@gmail.com,
Севрюкова В.В., студент, e-mail: se_vv@bk.ru,
ЮЗГУ, г. Курск, Российская Федерация

ФАЙЛОВЫЕ СИСТЕМЫ ДЛЯ НАКОПИТЕЛЕЙ НА ОСНОВЕ ФЛЭШ-ПАМЯТИ

В статье рассмотрен стандарт формата ELF, виды исполняемых файлов в Linux, а также основной функционал файловых систем для флэш-памяти, разработанных на базе процессорных модулей «Багет-83» и «Багет-83В».

***Ключевые слова**: флэш-память, перемещаемый файл, разделяемый объектный файл, исполняемый файл.*

FILE SYSTEMS FOR FLASH MEMORY DRIVES

The article discusses the ELF format standard, the types of executables in Linux, as well as the basic functionality of file systems for flash memory, developed on the basis of the Baget-83 and Baguette-83B processor modules.

***Key words**: flash memory, moveable file, shared object file, executable file.*

Для накопителей на основе флэш-памяти необходимо использование специализированных файловых систем, учитывающих характеристики и особенности работы с данными для флэш-памяти. С учётом данных особенностей и особенностей при разработке файловой системы в ОСРВ «Багет-83» и «Багет-83В» необходимо реализовать возможность нахождения образа по сигнатуре, проверки его целостности и передачи ему управления.

Файловая система – порядок, который определяет способ организации, хранения, именования данных на различных носителях информации. Она определяет размер имён файлов, максимальный возможный размер файла, набор атрибутов файла [1].

Файловые системы для флэш-памяти предоставляют следующий функционал, который отличается от традиционных файловых систем:

1. Организация хранения данных на флэш-памяти не требует сокращения времени на поиск данных на носителе;

2. Для блока флэш-памяти, предназначенного для записи данных, должна быть предварительно проведена процедура стирания данных;

3. Флэш-память имеет ограниченное количество допустимых циклов записи/стирания данных, поэтому файловые системы для флэш-памяти должны обеспечивать равномерный износ блоков [2].

ELF — формат двоичных файлов, используемый во многих современных UNIX-подобных операционных системах [3].

Стандарт формата ELF различает три типа файлов:

1. Перемещаемый файл;

2. Разделяемый объектный файл;

3. Исполняемый файл.

Перемещаемый файл содержит инструкции и данные, которые могут быть связаны с другими объектными файлами. К этому типу относятся объектные файлы статических библиотек.

Разделяемый объектный файл также содержит инструкции и данные и может быть связан с другими перемещаемыми файлами и разделяемыми объектными файлами, в результате чего будет создан новый объектный файл, либо при запуске программы на выполнение операционная система может динамически связать его с исполняемым файлом программы, в результате чего будет создан исполняемый образ программы.

Исполняемый файл содержит полное описание для создания образа процесса. В том числе: инструкции, данные, описание необходимых разделяемых объектных файлов и необходимую символьную и отладочную информацию.

Каждый исполняемый файл в Linux включает в себя следующие части:

1. Заголовок файла;

2. Таблица заголовка программы;

3. Таблица заголовков раздела;

4. Содержимое разделов и сегментов.

На рисунке 1 изображена структура исполняемого файла.

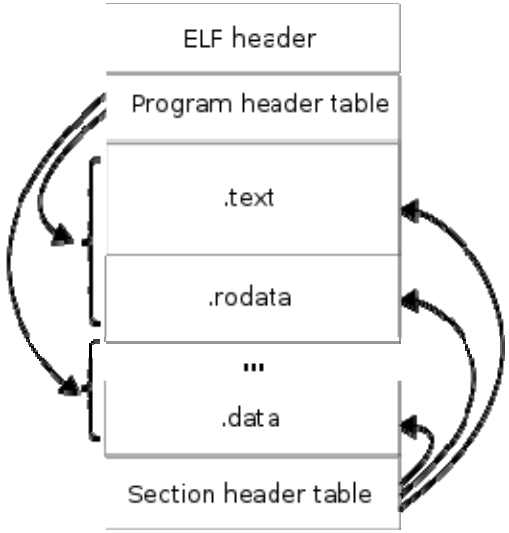


Рисунок 1 – Структура исполняемого файла

Заголовок файла имеет фиксированное расположение в начале файла и содержит общее описание структуры файла и его основные характеристики, такие как тип, версия формата, виртуальный адрес точки входа и др.

Таблица заголовков программы расположена сразу после заголовка файла и содержит заголовки сегментов, каждый из которых описывает отдельный сегмент программы и его атрибуты. Такими атрибутами могут быть:

1. Тип сегмента и действия операционной системы с данным сегментом;

2. Расположение сегмента;

3. Точка входа сегмента;

4. Размер сегмента;

5. Флаги доступа к сегменту (запись, чтение, выполнение).

Информация, содержащаяся в этой таблице указывает ядру системы, как собрать образ процесса из сегментов.

По таблице заголовков разделов можно найти расположение всех разделов в файле.

Сегменты содержат данные, необходимые для исполнения файла, а разделы – для линковки и обработки relocation. Каждый байт в файле может относиться не более чем к одной секции.

Список использованных источников

1. Беляев, М.А. Основы информатики [Текст]: учебник для вузов / М.А. Беляев, В.В. Лысенко, Л.А. Малинина. – Ростов-на-Дону: Феникс, 2006. – 352 с.

2. Грошев, А.С. Информатика [Текст]: учебник для вузов / А.С. Грошев. – Архангельск: Арханг. гос. техн. ун-т, 2010. – 470 с.

3. Робачевский, А.М. Операционная система UNIX [Текст] / А.М. Робачевский, С. А. Немнюгин, О. Л. Стесик. – СПб.: БХВ-Петербург, 2010. – 656 с.

Корсунский Н.А., студент, e-mail: cor.nick2013@yandex.ru
Макашин В.А., студент, e-mail: v.makashin@yandex.ru
ЮЗГУ, г. Курск, Российская Федерация

ПРИМЕНЕНИЕ НЕЧЕТКОГО МОДЕЛИРОВАНИЯ СЛОЖНЫХ ТЕХНИЧЕСКИХ СИСТЕМ

В статье рассматривается общее понятие теории нечетких множеств и практическое применение нечеткого моделирования сложных систем при проектировании современных технологий.

Ключевые слова: *теория нечетких множеств, нечеткая логика, модель, нечеткое множество, лингвистическая переменная, фуззификатор, цифровая обработка изображения, классификация сегментов изображения.*

APPLICATION OF FUZZY MODELING OF COMPLEX TECHNICAL SYSTEMS

The article discusses the general conception of fuzzy set theory and practical application of the fuzzy modeling of complex systems for the design of modern technologies.

Keywords: *fuzzy set theory, fuzzy logic, model, fuzzy set, linguistic variable, fuzzifier, digital image processing, division of image segments.*

Исследование окружающего мира и проектирование современных технологий невозможны без проведения разнообразных экспериментов. При этом далеко не всегда их можно провести, зачастую проведение таких экспериментов требует значительного времени, связано с риском и большими материальными издержками. В таких ситуациях используют математическое моделирование, основанное на проектировании моделей изучаемых объектов и процессов [1-3].

Основным требованием при разработке таких моделей является требование адекватности, то есть соответствие модели и изучаемого объекта по рассматриваемому явлению. Для многих технических систем и их элементов существуют точные модели, с помощью которых удается провести процесс проектирования без обращения к эксперименту над реальным объектом. Испытание таких технических моделей необходимо, в основном, для выявления производственных дефектов и ошибок. В сложных системах, где человек играет активную роль, действует принцип несовместимости: при исследовании поведения сложной системы необходимо отказаться от высокой точности и строгости, которые характерны для простых систем, и привлекать к ее анализу подходы, которые являются приближенными изучаемому объекту.

Неопределенность представлений человеческих знаний в разработке сложных математических моделей привели к необходимости создания теории нечетких понятий. Основоположником этой теории является американский математик Лотфи Заде, опубликовавший в 1965 году основы теории нечетких множеств, которое явилось результатом обобщения и переосмысления достижений в многозначной логике, теории вероятностей и математической статистики, дискретной математики, теории матриц, теории графов, теории грамматики.

С теорией нечетких множеств связано понятие лингвистической переменной, описывающее неточное (нечеткое) отражение человеком окружающего мира. Лингвистической переменной (linguistic variable) принято называть переменную, значениями которой могут быть слова или словосочетания некоторого естественного или искусственного языка. Используя это понятие, Л. Заде ввел определение нечеткого множества и построил

основы теории нечетких множеств, расширив одно из базовых понятий математики – понятия множества.

Нечетким множеством (fuzzy set) Ã на универсальном множестве U называется совокупность пар (μA(U), U), где μA(U) — степень принадлежности элемента u∈U к нечеткому множеству Ã. Степень принадлежности — это число из диапазона [0, 1]. Чем выше степень принадлежности, тем в большей мере элемент универсального множества соответствует свойствам нечеткого множества.

В основе понятия нечеткого множества лежит представление о том, что составляющие данное множество элементы, обладающие общим свойством, могут обладать этим свойством в различной степени и, следовательно, принадлежать к данному множеству с различной степенью. Модели и методы, использующие категорию нечеткости, очень важны при количественном анализе явлений, которые раньше либо могли быть учтены только на качественном уровне, либо требовали использования весьма грубых приближенных моделей [3,7,9].

Теория нечетких множеств составляет раздел теории нечеткой логики, которая наиболее близка к человеческому мышлению и естественным языкам, чем традиционные логические системы. Нечеткая логика, в основном, обеспечивает эффективные средства отображения неопределенностей и неточностей реального мира, позволяющие построить модель, адекватную реальности [8,10].

В настоящее время нечеткая логика рассматривается как наиболее успешный метод моделирования и проектирования. Системы на нечетких множествах используются в таких областях, как: медицинская диагностика, техническая диагностика, финансовый менеджмент, управление персоналом, распознавание образов, разведка ископаемых, управление компьютерными сетями, управление технологическими процессами, управление транспортом, логистика, поиск информации в Интернете, радиосвязь и телевидение.

Нечеткая логика, как модель человеческих мыслительных процессов, в настоящее время широко используется в системе искусственного интеллекта и в технологиях поддержки принятия решений. При решении задач обработки сложноструктурированных изображений используются методы, основанные на свойствах и операциях над нечеткими множествами. В качестве применения нечеткой логики в алгоритмах цифровой обработки

изображений рассмотрим задачу классификации цветных изображений мазков крови, воспользовавшись правилами нечеткого вывода [4-6].

Изображения препаратов крови, получаемые при проведении общеклинического обследования крови, представлены на рисунке 1.

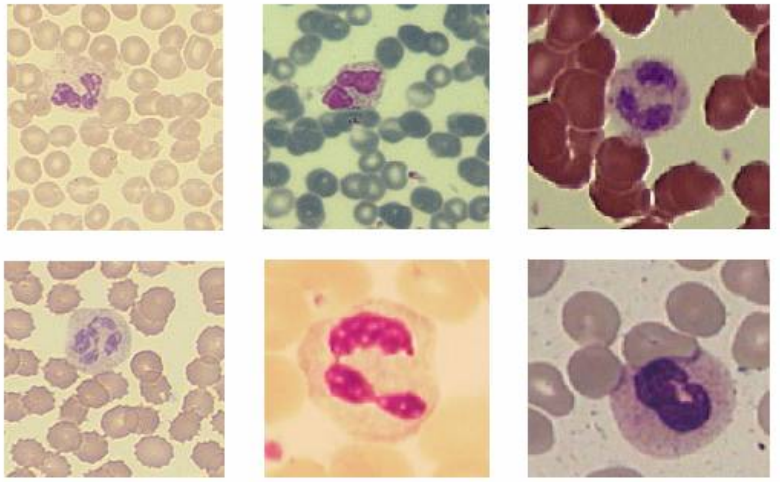


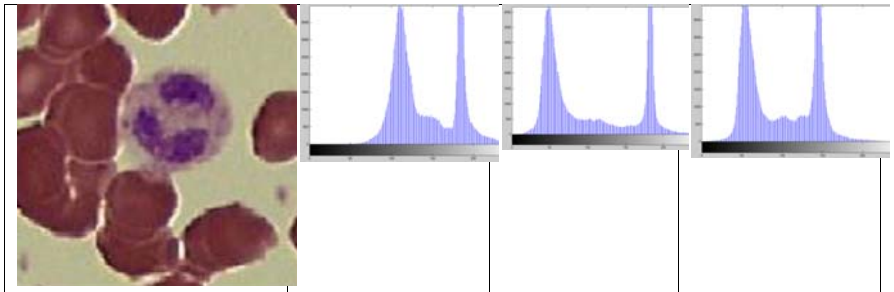
Рис. 1. Типы окрасов мазков

На изображениях присутствуют различные форменные элементы крови – клетки крови. На практике при обнаружении форменных элементов крови сталкиваются с основным и типичным для медицинских приложений компьютерного зрения препятствием – большой вариабельностью изображений, с которыми приходится иметь дело.

Форменные элементы крови могут быть классифицированы по двум независимым группам признаков. К первой группе относятся цветовые показатели [4-6].

База данных микрофотографий мазков имеет библиотеку стандартных окрасов мазков, взятых из практики работы различных лабораторий. В разработанной базе данных используются шесть окрасов, приведенных на рисунке 1.

Гистограммы RGB-кодов, полученные по этим окрасам изображений мазков, приведены на рисунке 2. Каждому окрасу мазка соответствует свой блок решающих правил.



а)исходное изображение	б) гистограмма R-канала	в) гистограмма G-канала	г) гистограмма B-канала

Рис. 2. Гистограммы средних RGB-кодов, полученные для различных окрасов изображений мазков

Так как изображения форменных элементов крови имеют неоднозначную окраску при любом типе окраса мазка, то о принадлежности сегмента изображения к определенному классу: эритроцит или лейкоцит – на основании мод RGB-кодов сегмента можно утверждать с определенной уверенностью, которая вычисляется на основе мод RGB-кодов сегмента и соответствующих нечетких решающих правил [7-10]. По гистограммам, приведенным на рисунке 2, строятся функции принадлежности, характеризующие уверенность принадлежности цветного изображения мазка периферической крови к соответствующему окрасу по данной цветной составляющей. На рисунке 3 представлены функции принадлежности для шести окрасов по коду R, построенные на основе гистограмм.

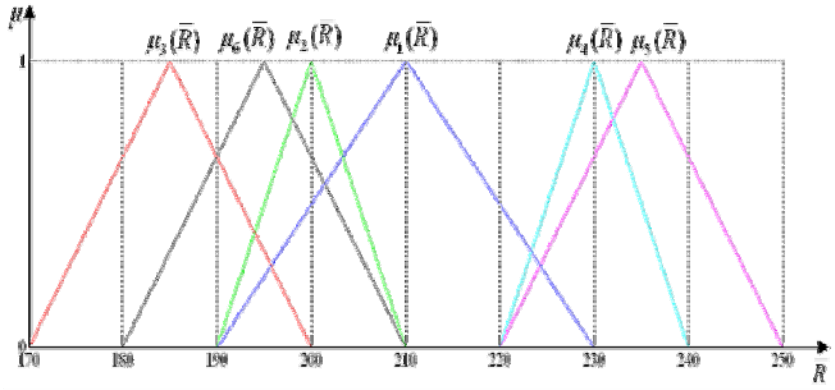


Рис. 3. Функции принадлежности к окрасам мазка по коду *R*

В соответствии с рисунком 3 определяется фуззификатор, который преобразует четкое значение RGB-кода в нечеткое – уверенность в принадлежности текущего изображения мазка к определенному окрасу.

В итоге для каждого окраса запишем соответствующую нечеткую продукцию, характеризующую уверенность принадлежности текущего изображения к одному из шести окрасов.

Например, чтобы определить уверенность принадлежности текущего изображения к четвертому окрасу, необходимо записать следующую нечеткую продукцию:

$$Если\ \mu_4(\overline{R})\ и\ \mu_4(\overline{G})\ и\ \mu_4(\overline{B}),\ то\ K_{y4}, \qquad (1)$$

где K_{y4} – коэффициент уверенности в принадлежности цветного изображения к окрасу четыре.

В нечеткой продукции (1), как правило, в качестве нечетких операций используются нечеткие *И*.

Окончательный окрас k_0 выбирается по максимальному коэффициенту уверенности, то есть

$$k_0 = \arg \max_{k=1,6}(K_{yk}).$$

По построенным гистограммам формируем фуззификатор для двух термов: эритроциты и лейкоциты, а затем формируем решающее правило:

$$Если\ \mu_э(\overline{R})\ и\ \mu_э(\overline{G})\ и\ \mu_э(\overline{B}),\ то\ K_y = эритроцит, \qquad (2)$$

$$Если\ \mu_л(\overline{R})\ и\ \mu_л(\overline{G})\ и\ \mu_л(\overline{B}),\ то\ K_y = лейкоцит. \qquad (3)$$

Для окончательного принятия решения снова воспользуемся

$$k_0 = \arg \max_{\xi=э,л}(K_{y\xi}), \qquad (4)$$

Вывод. Разработан модуль классификации сегментов цветного изображения, построенный на основе правил нечеткого вывода, анализирующих моды RGB-кодов как всего изображения, так и конкретных сегментов. Отличительная особенность предложенного метода заключается в том, что для классификации сегментов изображения используются две базы решающих правил, первая из которых определяет цветовой фон, к которому относится изображение, а вторая база – класс анализируемого сегмента, позволяющий определить коэффициенты уверенности в принадлежности анализируемого сегмента к одному из двух диагностируемых классов. В дальнейшем разработанный модуль будет реализован в системах автоматизированного анализа разнотипных признаков.

Список использованных источников

1. Томакова, Р.А. Структурно-функциональные решения нечетких нейронных сетей для интеллектуальных систем анализа разнотипных признаков/ Р.А. Томакова, С.А. Филист, В.В. Жилин, С.А. Горбатенко//Фундаментальные и прикладные проблемы техники и технологии.2011. №1. –С.85-91.

2. Томакова, Р.А. Теоретические основы и методы обработки и анализа микроскопических изображений биоматериалов: монография / Р.А. Томакова, С.Г. Емельянов, С.А. Филист; Юго-Зап. гос. ун-т. Курск, 2011. - 202с.

3. Томакова, Р.А. Математическое обеспечение распознавания и классификации сложноструктурируемых биологических объектов/ Р.А. Томакова, А.А. Насер, О.В. Шаталова// Международный журнал прикладных и фундаментальных исследований. 2012. №4. 48-29.

4. Брежнева, А.Н. Нейросетевые модели сегментации ангиограмм глазного дна на основе анализа RGB-кодов пикселей/ А.Н. Брежнева, С.А. Борисовский, Р.А. Томакова, С.А. Филист// Системный анализ и управление в биомедицинских системах. 2010.Т.9. №1. -.С.72

5. Томакова, Р.А. Способ сегментации ангиограмм глазного дна на основе нейросетевого анализа RGB-кодов пикселей/ Р.А. Томакова, А.Н. Брежнева, С.А. Филист //Известия ЮФУ. Технические науки. 2009. №9(98). –С.171-176.

6. Томакова, Р.А. Гибридные технологии в интеллектуальных системах идентификации лекарственных средств/ Р.А. Томакова, С.А. Филист, М.В. Томаков//Нейрокомпьютеры: разработка, применение. 2014.№6.- С.31-66.

7. Томакова, Р.А. Метод обработки сложноструктурируемых изображений на основе встроенных функций среды MATLAB/Р.А. Томакова, С.А. Филист//Вестник Забайкальского государственного университета. 2012. №1. –С.3-9.

8. Пихлап, С.В. Нечеткие нейросетевые структуры для сегментации изображений глазного дна/ С.В. Пихлап, Р.А. Томакова, С.А. Филист//Вестник Воронежского государственного технического университета. 2009. Т. 5. № 4. С. 42-45.

9. Томакова, Р.А. Нечеткие нейросетевые технологии для выделения сегментов с патологическими образованиями и морфологическими структурами на медицинских изображениях/ Р.А. Томакова, С.А. Филист, А.А. Насер // Биомедицинская радиоэлектроника.2012. №4. –С.43-50.

10. Филист, С.А. Метод классификации сложноструктурируемых изображений на основе самоорганизующихся нейронных сетевых структур/ С.А. Филист, Р.А. Томакова, О.В. Шаталова, А.А. Кузьмин// Радиопромышленность. 2016. №4. –С.57-65.

Астанина Т.М., студент, e-mail: tatiana_astanina@mail.ru
Терехов Д.А., студент, e-mail: Dimas_terex@mail.ru
ЮЗГУ, г. Курск, Российская Федерация

ПЕРСПЕКТИВЫ РАЗВИТИЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ, ПРЕДНАЗНАЧЕННЫХ ДЛЯ УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ В УСЛОВИЯХ ЦИФРОВОЙ ЭКОНОМИКИ

В статье рассматриваются этапы построения и развития автоматизированных систем управления для управления предприятием. Обосновываются критерии эффективности их функционирования. Рассматривается иерархическая декомпозиция частных математических моделей, позволяющих определить структуру системы управления, в которой функции управления распределены по уровням.

Ключевые слова: *автоматизированных систем управления, математические модели, условия декомпозиции, цели функционирования.*

PROSPECTS FOR THE DEVELOPMENT OF AUTOMATED SYSTEMS DESIGNED TO MANAGE THE ENTERPRISE IN A DIGITAL ECONOMY

The article deals with the stages of construction and development of automated control systems for enterprise management. Criteria of efficiency of their functioning are proved. The hierarchical decomposition of particular mathematical models allowing to define structure of control system in which control functions are distributed on levels is considered.

Key words: automated control systems, mathematical models, decomposition conditions, functioning purposes.

На сегодняшний день любая автоматизированная система управления предприятием (АСУП) представляет собой сложный многоуровневый комплекс, включающий программные, технические, информационные и организационные средства, а также человеческие ресурсы. АСУП предназначена для решения задач планирования и управления работой предприятия в различных сферах деятельности [1-3]. И, несмотря на то, что цель использования АСУП определена вполне точно и однозначно, специфика построения данных систем со временем изменяется, в зависимости от экономических условий в каждый конкретный период функционирования.

Появление в двадцатом веке первых АСУП стало совершенно закономерной и необходимой ступенью в развитии экономики. Объёмы данных хозяйствующих субъектов неуклонно росли, соответственно их сбор и обработка вручную становились всё более сложным, время затратным и трудоёмким процессом. Кроме того, эффективность управления производством не могла повыситься из-за нехватки вычислительных мощностей [4-6].

В пятидесятые годы двадцатого века в СССР началась разработка и внедрение автоматизированных систем управления. Тогда на продвижение в этой сфере влияло несколько факторов. В первую очередь, это направленность разработок исключительно на нужды военной сферы,а также высокая стоимость ЭВМ. Однако опыт, полученный при созданиимногочисленных вычислительных комплексов, специализированных ЭВМ, автоматизированных систем для противоракетной и противокосмической обороны, системуправления силами ВМФ и авиации, позволил в дальнейшем начать разработку АСУП. И уже в 1963-1964 годах были начаты работы по созданию АСУП на базе отечественных универсальных цифровых вычислительных машин. В СССР первой крупной АСУП стала система «Львов», внедренная на Львовском телевизионном заводе «Электрон».

Дальнейшее развитие АСУ заключалось в создании комплексных, интегрированных систем управления. Начиная со второй половины 70-х годов, для АСУП помимо автоматизации организационного управления предприятием, были определены новые задачи. В настоящее время АСУП стали применяться для технологической подготовки производства и автоматизированного проектирования новых и испытания готовых изделий. Также неотъемлемой составляющей развития АСУП являлось и совершенствование технической базы.

Со временем такая многофункциональность АСУП привела к появлению нового понятия ERP-системы – интегрированной системы планирования ресурсов предприятия, которое, будучи более точным и ёмким, практически вытеснило своего «предшественника».

В настоящее время ERP-системы продолжают развиваться и видоизменяться благодаря значительному продвижению в области моделирования и анализа бизнес-процессов предприятий, совершенствованию технической базы и благоприятным условиям рыночной экономики современной России. На рынке информационных технологий России автоматизированные системы управления представлены как многочисленными зарубежными разработками, так и широко известными отечественными программными средствами. В зависимости от специфики производства и масштабов предприятия некоторые организации не только покупают ПО, но и разрабатывают собственные ERP-системы, которые в большей степени отвечают требованиям конкретной организации.

Разработка автоматизированной системы начинается с оценки и прогноза состояния внешней среды [7,8].

Под внешней средой понимают совокупность управляемых объектов и объектов обслуживания, а под состоянием этой среды – динамические характеристики совокупности управляемых объектов и объектов обслуживания или управляемого процесса. Поскольку внешняя среда подвержена эволюции, необходимо прогнозировать ее развитие на период внедрения и жизненного цикла системы [9,10].

Далее переходят к формулировке целей создания системы. Одновременно с этим выбирают и обосновывают критерии эффективности и ограничения. На основе полученной информации разрабатывается общая математическая модель управления. Поскольку, как правило, автоматизированные системы создаются для реализации достаточно большого числа целей, возникает необходимость в разработке частных математических моделей управления конкретными объектами.

При разработке частных математических моделей должны соблюдаться основные условия иерархической декомпозиции:

- *согласованность уровней*;
- *подчиненность целевых функций низшего уровня целевым функциям высшего уровня;*
- *переход результатов решения задач высшего уровня в ограничения для задач низшего уровня*.

Частные математические модели являются основой для построения концептуальной модели, содержанием которой является перечень функций, выполняемых автоматизированной системой, и ее структура.

Иерархическая декомпозиция частных математических моделей позволяет определить структуру системы управления, в которой функции управления распределены по уровням.

Таким образом, на этом этапе производится определение функций, выполняемых системой, и разрабатывается ее общая структурная схема.

Этим заканчивается этап концептуального проектирования, результаты которого являются исходными данными для этапа логического проектирования.

В результате логического проектирования определяются: перечень задач, реализующих функции управления; граф информационно-логической взаимосвязи между задачами; совокупность алгоритмов, реализующих данные задачи; временные оценки реализации алгоритмов.

Системный подход предполагает, что разработка любой системы должна начинаться с определения цели, которой данная система должна достигнуть. Цель системы определяется факторами, внешними по отношению к данной системе. Отсюда следует, что определение цели предполагает проведение более общих исследований, связанных с анализом внешней среды [4-6].

Глобальная цель, поставленная перед системой, порождает в общем случае множество частных (локальных) целей, поскольку, как правило, глобальная цель непосредственно не достижима, и необходимо выделить ряд подчиненных ей частных целей, достигая которых можно прийти и к заданной общей цели. Локальные цели, таким образом, выступают как средства достижения глобальной цели.

Процесс формирования множества локальных целей не формализован. Одной и той же глобальной цели могут отвечать различные множества потенциальных локальных целей. Пока не выработана однозначная программа достижения главной цели, на множестве потенциальных локальных целей действует лишь отношение достаточности, когда можно утверждать, что некоторое подмножество целей достаточно для достижения данной цели. Исследование множества потенциальных целей, отсеивание и выбор, постепенное сужение круга рассматриваемых локальных целей, установление отношений порядка между различными локальными целями позволяет, в конце концов, сформировать упорядоченное множество целей, отвечающее вполне определенной программе достижения главной цели.

Процесс упорядочивания множества целей – важнейший этап проектирования системы, во многом определяющий ее структуру и общесистемные характеристики.

Системе может быть задано и несколько глобальных целей. В этом случае должен быть определен (или выбран на уровне проектирования) принцип компромисса. Цель обусловливает структуру и поведение системы.

Наличие цели – основополагающий принцип не только для процесса проектирования системы, но и для контроля функционирования созданной системы. Если система не обеспечивает реализации заданной цели или недостаточно своевременно реагирует на ее изменение, то можно считать, что система спроектирована неудачно, она должна быть либо модернизирована, либо заменена другой.

Требования, предъявляемые к целям:

– цель должна быть сформулирована так, чтобы ее можно было оценить (задать) количественно;

– в процессе проектирования должен быть предусмотрен некоторый «механизм», позволяющий оценить степень достижения заданной цели.

Из этих условий вытекает необходимость количественной оценки цели, для чего используются критерии эффективности.

Цель системы определяет ее назначение, смысл ее функционирования. Цель выражает точку зрения заказчиков или проектировщиков системы на то, для чего создается система, какие функции она должна выполнять.

В отличие от этого критерии эффективности позволяют определить, количественно хорошо или плохо работает система, насколько успешно она выполняет свои функции.

Критерий эффективности представляют собой количественную оценку того, как работает система [9,10].

Применительно к автоматизированным системам все множество критериев можно подразделить на три большие группы:

- интегральные, которые носят общий характер и дают оценку от внедрения системы за весь период ее службы.

Это, как правило, критерии экономической эффективности;

• дифференциальные, которые носят более локальный характер и предназначены для оценки работоспособности автоматизированной системы за определенный, но достаточно большой отрезок времени. Это так называемые критерии технической эффективности;

• точечные, характеризующие качество функционирования системы в данный момент времени. Это так называемые оперативные критерии качества управления. С их помощью вырабатываются решения по оптимизации процесса управления в каждый текущий момент времени.

Примерами критериев экономической эффективности могут быть: среднегодовой экономический доход за счет внедрения системы, суммарная стоимость системы, период абсолютной окупаемости системы, скорость окупаемости и т. п.

Однако у современных отечественных ERP-систем можно выделить существенный общий недостаток – низкий показатель оригинальности. Российские разработчики в основном ориентируются на достижения зарубежных коллег и создают системы, аналогичные уже существующим, в то время какболее дальновидным и правильным вариантом была бы генерация уникальных оригинальных идей и реализация собственного потенциала.

Список использованных источников

1. Юревич Е.И. Теория автоматического управления/ Е.И.Юревич — СПб.: «БХВ-Петербург». 2016. 551 с.

2.Томакова, Р.А. Структурно-функциональные решения нечетких нейронных сетей для интеллектуальных систем анализа разнотипных признаков/ Р.А. Томакова, С.А. Филист, В.В. Жилин, С.А. Горбатенко//Фундаментальные и прикладные проблемы техники и технологии.2011. №1. –С.85-91.

3. Леонтьева, Т.И. Автоматизированное проектирование женской одежды/ Т.И. Леонтьева, Т.М. Ноздрачева, Р.А. Томакова// Известия Юго-Западного государственного университета.2003. №1. – С.100-102.

4. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

5. Томакова, Р.А. Гибридные технологии в интеллектуальных системах идентификации лекарственных средств/ Р.А. Томакова, С.А. Филист,

М.В. Томаков//Нейрокомпьютеры: разработка, применение. 2014.№6.-С.31-66.

6. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/ А.В. Малышев// Известия Юго-Западного государственного университета.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

7. Томакова, Р.А. Структурно-функциональные решения нечетких нейронных сетей для интеллектуальных систем анализа разнотипных признаков/ Р.А. Томакова, С.А. Филист, В.В. Жилин, С.А. Горбатенко//Фундаментальные и прикладные проблемы техники и технологии.2011. №1. –С.85-91.

8. Филист, С.А. Метод классификации сложноструктурируемых изображений на основе самоорганизующихся нейронных сетевых структур/ С.А. Филист, Р.А. Томакова, О.В. Шаталова, А.А. Кузьмин// Радиопромышленность. 2016. №4. –С.57-65.

9. Чудинов, С.М. FPGA-технологии в автоматизированных системах для исследования изображений в форме флюорограмм/ С.М. Чудинов, Р.А. Томакова, В.А. Степанов, И.В. Зуев//Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2014. Т.29. №1-1(172). –С.7074.

10. Филист, С.А. Математическая модель системы автоматического регулирования давления в сердечно-сосудистой системе/ С.А. Филист,А.А. Кузьмин, Р.А. Томакова// Системный анализ и управление в биомедицинских системах. 2005.Т4.–№1. – С.50-53.

Драчев Е.А., студент, e-mail: kakos_mimino1@mail.ru
ЮЗГУ, г. Курск, Российская Федерация
Боков И.Н., студент, e-mail: Inbokov@gmail.com
ЮЗГУ, г. Курск, Российская Федерация

ПРОЕКТИРОВАНИЕ СЕРВИСОВ ДИНАМИЧЕСКОЙ ОТЛАДКИ ДЛЯ ОПЕРАЦИОННЫХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

В статье производится описание результатов разработки сервисов динамической отладки.

Ключевые слова: *программное обеспечение, ядро, сервер, аппаратно-независимая часть, «Монитор».*

DESIGN SERVICES DYNAMIC DEBUGGING FOR OPERATING SYSTEMS REAL-TIME

This article describes the results of developing dynamic debugging services.

Key words: *Software, core, server, hardware-independent part, "Monitor".*

В рамках работы по созданию сервисов динамической отладки была разработана архитектурно независимая часть (АНЗ) под названием «Монитор», позволяющая следить за состоянием процессов в операционных системах (ОС). «Монитор», в отличие от классических отладчиков, позволяет собирать информацию о процессах в рамках реального времени – без остановки системы в целом или ее определенного процесса в частности [1-3].

«Монитор» является расширяемым модулем, поддерживающим определенный сетевой протокол для внешнего взаимодействия и программный интерфейс для интеграции с новыми расширениями. Ядро «монитора» представляет собой сетевой сервер, который в рамках ОС является отдельным потоком ядра, осуществляющим обработку подключений и обмен данными с клиентом через расширения. Стоит отметить, что «монитор» не потребляет ресурсы ОС, если к нему не подключен клиент.

Собранные сервером данные сами по себе не имеют ценности. В рамках проведенных работ были созданы два клиентских приложения: для сбора данных и обработки. Программное обеспечение (ПО), предназначенное для сбора данных, позволяет собирать кадры, отправленные «монитором», и сохраняет их в бинарном виде в файловой системе, а также может расшифровать запакованные данные в бинарном виде и предоставить их в текстовом формате. Обработчик данных позволяет принимать на вход как сформированный заранее файл, так и сетевой поток данных в момент работы серверной части «монитора». Такой подход предоставляет возможность анализировать данные после завершения работы вычислителя на удаленном рабочем месте [4-7].

ПО клиента и сервера учитывает возможную разницу между ними в порядке байтов. Запуск клиента происходит из терминала linux, путем ввода команды «./client.elf 192.168.8.222», где ./client.elf – абсолютный путь к исполняемому файлу клиента, а 192.168.8.222 – IP-адрес сервера. Дополнительным параметром можно передать имя файла, в который будет сохранен лог работы клиента. В таком случае команда запуска будет иметь вид «./client.elf 192.168.8.222 log.bin», где log.bin – имя файла лога, который будет сгенерирован.

Важной особенностью АНЗ является то, что работа с клиентом происходит в интерактивном режиме. В этом процессе сначала клиент выводит прогресс подключения к серверу, возникшие ошибки или успешный статус подключения и настройки сервера. После этого он начинает выводить данные, полученные от сервера, в удобном для чтения и понимания пользователем формате. Клиент имеет консоль для интерактивной настройки клиента. Для входа в консоль необходимо нажать клавишу Enter. Далее можно ввести одну из команд поддерживаемых клиентом:

1) q – выход из программы;
2) c – продолжение работы клиента без изменений;
3) h – вывести справку по использованию команд;
4) t diff_time – установить время отправки кадров сервером;
5) v pidaddrsize – установить область памяти указанного процесса для слежения.

Выход из клиента происходит либо командой q, либо нажатием сочетания клавиш Ctrl+D, которое закрывает стандартный поток ввода.

Кроме интерактивной работы с клиентом имеется возможность получения информации о динамической отладке из файла лога, который задается дополнительным параметром клиента. Для запуска анализатора лога необходимо выполнить команду «./parser.elf<log.bin», где ./parser.elf – абсолютный путь к исполняемому файлу анализатора, а log.bin – путь к файлу лога, сформированного клиентом. В рабочей директории будут соз-

даны файлы, хранящие информацию о работе динамического отладчика, для модуля слежения за переменными будет создан текстовый файл «var.log»; для модуля слежения за переключениями задач будет создан текстовый файл «proc.log».

Для связи клиента и сервера используется сеть Ethernet, следовательно, для работы «монитора» вычислитель-сервер и ПЭВМ-клиент должны находиться в одной подсети. Протокол прикладного уровня «монитора» построен поверх протокола транспортного уровня TCP, обеспечивающего гарантированную доставку.

При установке соединения сервер отправляет клиенту нуль-терминальную строку с названием сервера. В ответ на нее клиент отправляет серверу магическое число (4 байта) и версию протокола (4 байта). Значение магического числа – 0xDEADFACE, текущее значение версии протокола – 1. В ответ сервер присылает флаги поддерживаемых модулей (4 байта). На данный момент поддерживаются 2 модуля: модуль слежения за переменными и модуль слежения за переключением процессов. Далее клиент отсылает серверу пакет, содержащий настройки для дальнейшей работы сервера.

В процессе работы клиент и сервер могут обмениваться специальными пакетами. Клиент может отправлять серверу только пакеты настроек.

Пакет настроек имеет следующую последовательную структуру:

1) 4 байта – номер модуля, для которого предназначена конфигурация (0xff – настройки для самого сервера).

2) 4 байта – размер конфигурации для указанного модуля (8 байт для сервера).

3) Данные для конфигурации модуля.

3.1) Данные конфигурации для сервера:

1) 4 байта – время между пересылаемыми кадрами в мкс;

2) 4 байта – используемые модули.

3.2) Данные конфигурации для модуля слежения за памятью:

1) 8 байт – количество областей для слежения.

3.3) Массив областей памяти:

1) 8 байт – идентификатор процесса, за областью памяти которого мы будем следить;

2) 8 байт – адрес памяти, за которой будем следить;

3) 8 байт – размер памяти, за которой будем следить.

4) Пакеты идущие от сервера пользователю. От сервера могут идти пакеты двух типов: информационные и пакеты данных.

4.1) 4 байта – тип пакета(0 – информационный, 1 – данные).

4.2) Структура пакета данных:

1) 8 байт – время формирования пакета (нс с момента запуска ОС4000);

2) 4 байта – флаги (от каких модулей отладчика пришла информация).

4.3) Если установлен флаг модуля переключения процессов, то дальше идет:

1) 8 байт – количество переключений процессов.

4.4) Массив переключений процессов:

1) 8 байт – идентификатор процесса, который был установлен;

2) 8 байт – идентификатор потока, который был установлен;

3) N байт – текстовое имя потока, в формате «паскаль строки»;

4) 8 байт – время переключения потока/процесса.

4.5) Если установлен флаг модуля отслеживания памяти, то дальше идет:

1) 8 байт – количество измененных переменных.

4.6) Массив измененных переменных:

1) 8 байт – идентификатор процесса, память которого была изменена;

2) 8 байт – адрес памяти, которая изменилась;

3) 8 байт – размер памяти, которая изменилась;

4) N байт – измененные данные (после изменения).

4.7) Структура информационного пакета:

1) 1 байт – модуль, от которого пришло информационное сообщение;

2) 2 байта – тип информационного сообщения (1 – просто информация, 2 – успешная операция, 3 – требуется переподключение, 4 – произошла ошибка на стороне клиента, 5 – произошла ошибка на стороне сервера);

3) 1 байт – код сообщения (зависит от типа);

4) N байт – текстовое описание в формате «паскаль строки».

Для создания расширения необходимо определить пять функций: инициализация, включение, выключение, реконфигурация и отправка; указать флаг расширения и модифицировать структуру, хранящую список доступных модулей. Каждая из функций вызывается либо периодически, либо по запросу от клиента.

Функция инициализации должна иметь прототип T_RETURN init(T_UINT32 version). Эта функция вызывается один раз при включении «монитора». В ее задачи входит задание конфигурации модуля по умолчанию, выделение памяти под требуемые структуры модуля, начальная инициализация переменных модуля и проверка условий необходимых для работы модуля. По завершении этой функции модуль должен перейти в состояние «готов к запуску». Функция возвращает 0 в случае успеха, или -1 – в случае ошибки.

Функция включения должна иметь прототип T_VOID enable(T_VOID). Она вызывается в случае, когда пользователь прислал запрос на включение данного модуля, если до этого модуль был выключен. Функция должна произвести включение модуля: запуск необходимых потоков, установка нужных флагов, определение параметров времени и т. п. Функция не должна завершаться с ошибкой (все условия возникновения ошибок должны были провериться функцией инициализации).

Функция выключения должна иметь прототип T_VOID switchDisable(T_VOID). Она вызывается в случае, когда пользователь прислал запрос на выключение данного модуля, если до этого модуль был включен. Функция должна произвести остановку работы модуля (остановить потоки, очистить буфера, сбросить таймеры и т. п.). Функция не должна завершиться с ошибкой.

Функция конфигурации должна иметь прототип T_RETURN reconfig(T_CONST_STRING buf, T_UINT32 size). Функция может вызывается при запросе пользователя на реконфигурацию модуля. Эта функция может быть вызвана как при выключенном состоянии модуля, так и при включенном. Функция может вызываться не ограниченное количество раз. Она должна обеспечивать смену настроек модуля(установку/снятие таймеров, адресов памяти и т. п.). Возвращает 0 в случае успеха и -1 в случае ошибки.

Функция отправки должна иметь прототип T_VOID monitorProcSwitchSend(T_VOID) и вызывается с определенной периодичностью, если только модуль был включен. Функция должна произвести отправку данных модуля клиенту, используя интерфейсные функции «монитора» (sendUint64, sendUint32, sendUint8, sendRawData, sendStatus, ntohll, htonll, ntohllFromBuf, ntohlFromBuf).

Флаг расширения задается в перечислимом типе T_MONITOR_MODULE_TYPE. Значение флага не должно использоваться другими модулями. Флаг может принимать значения в диапазоне от 0 до 31.

Для подключения необходимо указать модуль в списке доступных модулей, который описывается внутренней переменной modules в главном файле сервера. Переменная modules является массивом структур, где индекс массива должен соответствовать флагу модуля. Структура имеет шесть полей: первое поле является логической переменной (TRUE – модуль используется, FALSE – модуль не может быть использован), остальные пять полей структуры – это указатели на функции определенного модуля.

В стандартном комплекте вместе с «монитором» поставляются два расширения: для сбора информации о переключении разделов и процессов внутри разделов и для опроса значений ячеек памяти, выбранного диапазона адресов с заданной частотой.

Расширение для сбора информации о переключении процессов внутри раздела, а так же самих разделов, предоставляет информацию о том, какой процесс или раздел работал, и какой начал работать. Кроме этого предоставляется информация о времени переключении раздела или процесса. Это позволяет проверить правильность работы расписания разделов, а так же понять, каким образом переключаются процессы внутри раздела, сколько времени работал тот или иной процесс/раздел, в какой момент произошло переключение, и какой процесс/раздел был выбран следующим [7-10].

Вывод. Таким образом, расширение для опроса значений ячеек памяти выбранного диапазона с заданной частотой позволяет следить за изменением переменных, буферов и других участков памяти указанного раздела. В совокупности с предыдущим модулем можно получить информацию, какой процесс изменил указанный диапазон памяти.

Разработанный модуль «Монитор» предназначен для отслеживания состояния процессов в операционных системах.

Список использованных источников

1. Гордеев, А.В. Операционные системы: Учебник для вузов. 2-е изд. /А.В. Гордеев. – СПб.: Питер, 2006. – 416 с.;

2. Олифер, В.Г. Сетевые операционные системы/ В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2009. – 672 с.

3. Фонин Ю.Н., Грассман С. Архитектура и принципы построения операционной среды «мини-ОС», Труды Института Системного Программирования РАН, 2004, С. 159-166.

4.Малышев, А.В. Параллельный алгоритм реконфигурации логической структуры матричного мультипроцессора/ А.В. Малышев, А.В. Апальков// Глобальный научный потенциал.2012. № 20. С.108-110.

5. Томакова Р.А., Апальков В.В. Методы и алгоритмы теории принятия решений. - Курск, 2015. - 164 с.

6. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

7. Аникина Е.И. Информатика: Адаптационный курс: учеб. пособие / Е. И. Аникина, Е. В. Павлова; Юго-Зап. гос. Ун-т. Курск, 2016.- 95 с.

8. Томакова, Р.А. Метод обработки сложноструктурируемых изображений на основе встроенных функций среды MATLAB/Р.А. Томакова, С.А. Филист//Вестник Забайкальского государственного университета. 2012. №1. –С.3-9.

9. Томакова, Р.А. Математическое обеспечение распознавания и классификации сложноструктурируемых биологических объектов/ Р.А. Томакова, А.А. Насер, О.В. Шаталова// Международный журнал прикладных и фундаментальных исследований. 2012. №4. 48-29.

10. Томакова, Р.А. Гибридные методы и алгоритмы для интеллектуальных систем классификации соложноструктурируемых изображений: автореф.дис. … докт.техн.наук: 05.13.17/Томакова Римма Александровна. – Белгород, 2013. –42с.

Емельянова Е.С., студент, , e-mail: eemelyanova12@yandex.ru
Шепелева Ж.А., студент, e-mail:schepelewa.janna@yandex.ru
Аболмасова Е.С., студент, e-mail:evgeniya.abolmasova@gmail.com
ЮЗГУ, г. Курск, Российская Федерация

АНАЛИЗ РАЗВИТИЯ ВЕБ-СЕРВЕРОВ И ВЕБ-ТЕХНОЛОГИЙ

В статье рассмотрена история таких программных компонентов как сервер, а также различные подходы к веб-программированию. Описаны их достоинства и недостатки.

Ключевые слова: *Интернет, веб-программирование, сервер, Apache, Ajax, Node.js.*

ANALYSIS OF THE DEVELOPMENT WEB SERVERS AND WEB TECHNOLOGIES

The article describes the history of such software components as the server, as well as various approaches to web programming. Their advantages and disadvantages are described.

Key words*: Internet, web programming, server, Apache, Ajax, Node.js.*

Стремительное развитие сети Интернет за последние несколько лет привело к увеличению количества новых онлайн-сервисов. Интернет развился в крупную глобальную сеть с большим количеством различных сервисов и технологических решений, а также стал местом общения бизнес-организаций практически для всего мирового сообщества. В этом отношении возможности, предоставляемые Интернетом в современной деловой деятельности несравнимы ни с одним из решений, которые были представлены ранее. При этом Интернет-пространство создает условия одинаковые для всех деловых людей, вне зависимости от того, в каком уголке мира они находятся, и обеспечивает возможности для их успешной презентации. Это именно эти возможности обусловили причину появления новых интернет-продуктов, оригинальных программных решений, веб-инструментов и технологий.

Быстрое развитие сети Интернета также неизбежно ведет к поиску новых компетенций и возникновению новых сфер деятельности, направлений исследований [1-3]. В настоящее время веб-программисты повсеместно

востребованы обществом, поскольку являются профессионалами в области информационных технологий, осуществляющие разработку веб-приложений и создающие качественные и функциональные сайты. Важным аспектом для успешной работы с веб-технологиями является правильный выбор языка веб-программирования: клиентского и серверного. Программы, реализованные на клиентских языках, выполняет браузер и обрабатываются они на стороне пользователя. Вследствие этого одной из проблем клиентских языков является зависимость результата выполнения программы от браузера пользователя [[4-6]. Следует отметить, что в разных браузерах или в различных версиях одного и того же браузера одна и та же программа будет выполняться по-разному. Этот недостаток устраняется с помощью использования серверных языков, например, таких как PHP, Python, Ruby, Java [7-9].

Важной стороной работы серверных языков является возможность организации непосредственного взаимодействия с системой управления базами данных, сервером базы данных, упорядоченно хранящих информацию, которая может быть вызвана в любой момент. Сервер представляет собой программный компонент вычислительной системы, выполняющий обслуживающие функции по различным запросам клиентов, предоставляя им доступ к определённым ресурсам или услугам. С целью установления связи с клиентом, сервер выделяет необходимые ресурсы для межпроцессного взаимодействия и ожидает запросы на открытие соединения. В зависимости от типа такого ресурса, сервер может обслуживать процессы в пределах одной компьютерной системы или процессы на других машинах через каналы передачи данных или сетевые соединения. Следует заметить, что формат запросов клиента и ответов сервера обусловливается определенным видом протокола.

В зависимости от вида выполняемых задач одни типы серверов могут простаивать в ожидании, при отсутствии запросов на обслуживание, в то время как другие выполняют различные виды работ, например, по сбору информации, поэтому для таких серверов работа с клиентами может быть второстепенной задачей. В 1995 году сформировалась группа разработчиков, выпустивших первую версию сервера Apache (сокращение от "a patchyserver").

В разработанных и применяемых в настоящее время системах используются две версии Apache — 1.3 и Apache — 2.0. Обе версии широко представлены и занимают половину серверного рынка в мире. Основная

причиной успеха Apache заключается в широком спектре его функциональных возможностей. Следует заметить, что Apache может обслуживать одновременно большое количество клиентов, легко настраивается с помощью текстовых конфигурационных файлов и может быть переконфигурирован в процессе использования. Для разработки модулей имеется хорошо документированное API.

Использование скриптовых языков позволяет использовать Apache в связке с базами данных и серверами приложений [7-9]. В настоящее время разработан новый подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в обмене данными браузера с веб-сервером– Ajax.

Отличительной особенностью этого подхода является тот факт, что в результате обновления данных веб-страница не перезагружается полностью, но при этом веб-приложения работают быстрее и удобнее. При использовании Ajax пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент, далее скрипт, написанный на языке JavaScript определяет, какая информация необходима для обновления страницы. При этом браузер отправляет соответствующий запрос на сервер, который в свою очередь возвращает только ту часть документа, на которую пришёл запрос. Затем скрипт вносит изменения, с учетом полученной информации, без полной перезагрузки страницы, что позволяет существенно экономить время. В то время как в классической модели веб-приложения в ответ сервер генерирует совершенно новую веб-страницу и браузер полностью перегружает всю страницу.

В последнее время стремительно растет интерес к серверной реализации языка программирования JavaScript, основанной на движке V8 от Google — Node или Node.js, разработанной в 2009 году. До этого в серверах выполнялся подход «один поток на каждое соединение», а Даль придумал использовать систему, которая ориентирована на события. То есть реагирует на действие или бездействие и выделяет под это ресурс. Node.js предназначен для создания масштабируемых распределённых сетевых приложений, таких как веб-сервер. JavaScript выполняет действие на стороне клиента, а Node— на сервере.

Следует отметить, что Node.js позволяет добавление новых возможностей, таких как: подключение других внешних библиотек, написанных на разных языках и обеспечивает вызовы к ним из JavaScript-кода. Други-

ми словами, Node.js это программная платформа, среда выполнения JavaScript, точно так же как браузер.

Вывод: Таким образом, можно сделать вывод, что с Node проще производить масштабирование. При одновременном подключении к серверу большого количества пользователей (сотен тысяч) Node работает асинхронно, то есть ставит приоритеты и, вследствие этого, оптимально распределяет ресурсы. В то время как, например, Java на каждое подключение выделяет отдельный поток, что, несомненно, приводит к задержкам.

Список использованных источников

1. Хокинс, С. Администрирование Web-сервера Apache и руководство по электронной коммерции. / С. Хокинс. – Вильямс. 2001. – 336 с.

2. Крейн, Д. Ajax в действии. / Д. Крейн, Э. Паскарелло, Д. Джеймс. – Вильямс. 2008. – 640.

3. Пауэрс, Ш. Изучаем Node.js. / Ш.Пауэрс – СПБ.: Питер. 2014. – 400с.

4. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап. гос. ун-т. – Курск, 2018. –258с.

5. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/ А.В. Малышев// Известия Юго-Западного государственного университета.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

6. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

7. Томакова, Р.А. Интеллектуальные технологии сегментации и классификации биомедицинских изображений: монография/ Р.А. Томакова, С.Г. Емельянов, С.А. Филист; Юго-Зап. гос. ун-т. Курск, 2012. -222 с.

8. Томакова, Р.А.Построение системы распознавания образов на основе реализации метода потенциальных функций/ Р.А. Томакова, А.Н. Брежнева, Н.А. Корсунский// Мониторинг. Наука и технологии. 2017. №2(31). –С. 46-50.

9. Харзеева С.Э., Лушникова Е.И. Контекстуальное смысловое моделирование научного дискурса//Русский язык: исторические судьбы и современность . IV Международный конгресс исследователей русского языка: труды и материалы.-М.- 2010. -С. 144-145.

Жерденко К.А., студент, karina.gerdenko@rambler.ru
Мисинева Т.В., студент, tanyaspartak@gmail.com
ЮЗГУ, г. Курск, Российская Федерация

ИСПОЛЬЗОВАНИЕ МЕТОДА ЧЕТНОЙ АДРЕСАЦИИ ДЛЯ ДЕКОМПОЗИЦИИ РАСТРОВЫХ ИЗОБРАЖЕНИЙ В ПРОСТРАНСТВЕННЫХ БАЗАХ ДАННЫХ

В данной статье описывается метод, позволяющий выполнять выделение объектов на растровых изображениях с последующим занесением этих объектов в пространственную базу данных, а также результат применения описываемого метода.

Ключевые слова: *пространственные базы данных, растровое изображение, пиксель, объект, маска.*

USE OF THE FREQUENCY ADDRESSING METHOD FOR DECOMPOSITION OF RASTER IMAGES IN A SPATIAL DATABASE

This article describes a method that allows to select objects on raster images and record these objects into a spatial database. It also demonstrates result of applying of chosen method.

Keywords: *spatial databases, a raster image, a pixel, an object, a mask.*

В настоящее время во многих сферах человеческой деятельности используются пространственные базы данных. Они помогают хранить необходимую геолокационную информацию о товарах, посылках, транспортных узлах и т.д. В таких базах хранятся данные, которые представляются обычным пользователям в виде карт, отображающих информацию об определенной местности [1-3]. Карта – своего рода многослойная фотография области, где каждый слой представляет собой какой-либо объект на ней. В этом случае возникает проблема хранения растровой информации, принадлежащей разбитому на объекты изображению-карте. Для осуществления разбиения изображений на слои был выбран метод четной адресации [4] с программной реализацией на языке JavaScript, в качестве СУБД использовалась Oracle.

При реализации данного метода таблица базы данных, использующаяся для хранения объектов изображения, должна иметь три столбца: но-

мер объекта изображения, адрес начального пикселя объекта текущей серии, маска адреса [5-6]. Адрес пикселя представляется в двоичной системе счисления. Маска адреса необходима для определения множества адресов пикселей, принадлежащих текущему объекту изображения, и может содержать «0» или «1». В бит маски записывается «1» тогда, когда бит адреса пикселя совпадает с битом адреса множества пикселей, и «0», когда данный бит может принимать любое значение.

Алгоритм работы метода четной адресации основывается на группировке множества подряд идущих пикселей за счет четности адреса начального пикселя серии [7]. Он заключается в следующем. Программа находит пиксель объекта изображения и считывает его адрес. Если адрес нечетный, то это значит, что множество пикселей текущей серии состоит из одного пикселя, и адрес этого пикселя записывается в базу данных, а маска будет состоять из всех единиц. Если адрес пикселя четный, т.е. заканчивается на «0», то это значит, что множество пикселей серии будет содержать больше одного пикселя.

Приведем пример. Программа считала пиксель с адресом «10111000». Так как адрес пикселя четный, нам необходимо определить, сколько пикселей он может «вместить» в текущую серию адреса, а также посчитать количество пикселей, идущих подряд за найденным (далее «серия пикселей объекта»). Адрес «10111000» заканчивается на три нуля, значит, текущая серия может состоять из $2^3 = 8$ адресов последующих пикселей. Далее программа принимает решение из трех вариантов в зависимости от количества подряд идущих пикселей.

1. Если длина серии пикселей объекта равна количеству пикселей, способных «поместиться» в адрес текущего пикселя (текущую серию адреса), то в базу данных будет внесена информация об адресе текущего пикселя и маска вида «11111000» [8-9].

2. Если длина серии пикселей объекта меньше, то программа повторяет аналогичные действия, только адрес текущего пикселя будет рассматриваться так, как будто у него на один «0» в адресе стало меньше (т.е. в данный момент в адресе «10111000» рассматривались три последних нуля, теперь же программой будут рассматриваться два последних нуля).

3. Если длина серии пикселей объекта больше, то программа запишет максимально возможную серию адресов пикселей объекта в базу данных, а далее будет рассматривать пиксель, следующий за адресом последнего внесенного пикселя в базу данных [10].

В процессе исследования было использовано изображение, представленное на рисунке 1 (слева). Данное изображение имеет формат «.bmp», размер 1024 на 1024 пикселей и содержит 89 объектов. На рисунке 1 (справа) представлен объект под номером «1», являющийся частью экспериментального изображения.

Рис. 1. Экспериментальное изображение и объект «1»

На рисунке 2 показан пример таблицы, содержащей данные об объекте под номером «1» на экспериментальном изображении, где столбец «obj_id» обозначает номер объекта, столбец «addr» обозначает адрес объекта в двоичном представлении, а столбец «mask» обозначает маску для адреса объекта.

obj_id	addr	mask
1	01000100001010110000	11111111111111111000
1	01000100011010101100	11111111111111111100
1	01000100011010110000	11111111111111111000
1	01000100011010111000	11111111111111111110
1	01000100101010101100	11111111111111111100
1	01000100101010110000	11111111111111111000

Рис. 2. Пример таблицы с занесенными данными об объекте «1» на экспериментальном изображении

В процессе тестирования метода на представленных выше данных были получены результаты, приведенные в таблице 1.

Таблица 1 – Результат применения метода четной адресации

Название метода	Скорость внесение информации об изображении, сек	Количество занесенных записей в таблицу базы данных
Метод четной адресации	90,2	134979

Таким образом, в результате исследования была выполнена программная реализация метода четной адресации, позволяющая производить выделение объектов растрового изображения с последующей их записью в пространственную базу данных.

Список использованных источников

1. Атакищев О.И. Отображение графической и атрибутивной информации фрагментов изображения, представленных линейными квадродеревьями, на основе операция реляционной алгебры [Текст] / О.И. Атакищев, А.В. Белов, В.Г. Белов / Наукоемкие технологии. 2012. Т. 13. № 9. С. 34-37.

2. Белов А.В. Представление квадродеревьев бинарными деревьями [Текст] / А.В. Белов, Т.М. Белова / Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. № 1. С. 12-15.

3. Белов А.В. Способы хранения растровых данных на основе квадродеревьев в системах поддержки принятия решений [Текст] / А.В. Белов, Т.М. Белова // Известия Юго-Западного государственного университета. 2012. № 4-2 (43). С. 84-87.

4. Белова Т.М. Структура программы для представления алгоритмов управления процессом тестирования с помощью структуры данных [Текст]/ Т.М. Белова, В.Г. Белов, К.А. Жерденко // Информационные системы и технологии: материалы докладов II Международной научно-практической заочной конференции «ИСТ -2016». – Курск, ЗАО «Университетская книга», 2016. – С. 52 -54.

5. Белов В.Г. Представление пространственных объектов отрезками кривых, заполняющих растровое пространство [Текст] / В.Г. Белов, Т.М. Белова // Интеллектуальные информационные системы: тенденции, проблемы, перспективы. Материалы докладов IV международной заочной научно-практической конференции «ИИС-2016» (20 января 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 8 -11.

6. Белова Т.М. Представление параллельных и асинхронных алгоритмов в виде структур данных [Текст] / Т.М. Белова, Е.С. Кофанова, А.С. Тулупцева // Интеллектуальные информационные системы: тенденции, проблемы, перспективы. Материалы докладов IV международной заочной научно- практической конференции «ИИС-2016» (20 января 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 11 -12.

7. Белов В.Г. Способ кодирования для растровой формы представления пространственных объектов [Текст] / В.Г. Белов, Т.М. Белова // Оптико- электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации. Распознавание 2017. Сборник материалов XIII Международной научно-технической конференции (16 – 19 мая 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 63- 64.

8. Атакищев, О.И. Трехуровневая объектно-ориентированная модель организации параллельных асинхронных вычислительных процессов в ГИС [Текст] / О.И. Атакищев, Т.М. Белова, М.В. Белов // Известия Курск. гос. техн. ун-та. - 2004. - №2(13). - С. 67-72.

9. Белов В.Г. Определение пересечения пространственных объектов, представленных в растровой форме, с помощью модифицированных B PLUS деревьев [Текст] / В.Г. Белов, Т.М. Белова // Информационные системы и технологии. Сборник материалов III Международной научно-технической конференции. – Курск: ЗАО «Университетская книга», 2017. – С. 56 - 58.

10. Белов В.Г. Определение пересечения пространственных объектов, представленных в растровой форме, с помощью операции естественного соединения [Текст] / В.Г. Белов, Т.М. Белова // Инфотелекоммуникации и космические технологии: состояние, проблемы и пути решения: сборник научных статей по материалам I Всероссийской науч.-практ. конф.: в 2 ч. – Ч. 1 / редкол.: В. Г. Андронов (отв. ред.) [и др.]; Юго-Зап. гос. ун-т. – Курск, 2017. – С. 333 – 335.

Жерденко К.А., студент, e-mail: karina.gerdenko@rambler.ru
Малышев А.В., доцент, к.т.н., e-mail: alta76@yandex.ru
ЮЗГУ, г. Курск, Российская Федерация

ПРИМЕНЕНИЯ MVC-ПАТТЕРНА ВО FRONT-END РАЗРАБОТКЕ ВЕБ-ПРОГРАММ

В статье рассмотрены современные подходы к разработке клиентской части веб-приложений. В данный момент огромное влияние в мире веб-программирования имеют паттерны проектирования. Особой популярностью пользуется паттерн MVC, который используется не только при написании серверного ядра программы, но и при программировании клиентской части. Он позволяет облегчить разработку интерфейса, сделать систему более гибкой и устойчивой к изменениям, а также динамически изменять данные. В качестве примера применения данного паттерна рассмотрен JavaScript-фреймворк AngularJS, который используется для создания клиентской логики приложения.

Ключевые слова: *паттерн проектирования, шаблоны проектирования, объектно-ориентированное программирование, веб-приложение, front-end разработка, MVC (Model-View-Controller), JavaScript-фреймворк, AngularJS.*

FRONT-END WEB DEVELOPMENT WITH USING AN ARCHITECTURAL PATTERN CALLED MVC

This article presents the modern bases of front-end web development and explains the influence of using an architectural pattern MVC. This pattern is used in interface development and helps to make the whole system more flexible and stable. There is the example of the pattern MVC that's called AngularJS.

Keywords: *an architectural pattern, object-oriented programming, front-end web development, MVC (Model-View-Controller), AngularJS.*

В настоящее время широко используется в программировании паттерны для проектирования, которые позволяют облегчить не только чтение кода, но и его дальнейшую поддержку.

Одним из самых популярных паттернов в мире веб-программирования является паттерн MVC. Он используется при создании ядра веб-

приложения для короля делегирования запросов к сайту и проецируется на серверные языки, помогая сделать систему гибкой и устойчивой к изменениям [1, 2].

Однако в данный момент набирает популярность тенденция внедрения данного паттерна в клиентскую часть веб-приложений. В первую очередь это связано с необходимостью отделять написание представления (внешнего вида сайта, html-разметки) от вспомогательного кода (например, php- и js-скрипты). Такое разделение помогает сделать код «чистым», более читабельным, а также инкапсулировать его отдельные участки.

Во-вторых, внедрение данного паттерна позволяет адаптировать интерфейс к постороннему коду. Нет необходимости переписывать представление из-за изменения модели, ведь оно не зависит напрямую от данных.

Ещё одним преимуществом использования данного паттерна во front-end разработке является факт упрощения тестирования модулей программы. Так, тестирование интерфейса может производиться отдельно от взаимодействующих с ним контроллеров и моделей, а также независимо от представления данных в системе.

В связи с этим веб-разработчики задумались о целесообразности внедрения во front-end системы готовых модулей, предоставляющих возможность проектировать логику приложения с использованием паттерна MVC [3 - 5].

Одним из ярких примеров реализации данного паттерна является JavaScript-фреймворк от компании Google под названием AngularJS. Изначально этот фреймворк предполагалось использовать в одностраничных приложениях из-за его удобства и быстроты реагирования. Однако вскоре он нашел свое применение в больших веб-проектах, так как значительно упрощал процесс разработки и повышал её скорость.

AngularJS имеет ряд преимуществ, объясняющих его использование.

Во-первых, это манипуляция данными в режиме реального времени. Связывание данных и внедрения зависимостей устраняет большую часть кода. Благодаря ей фреймворк позволяет:

- контролировать действия пользователя на странице;
- осуществлять двухстороннюю привязку данных, обеспечивающую их синхронизацию;
- динамически обновлять данные при изменении состояния системы или после получения команды от сервера;
- управлять данными, отправляя запросы на сервер;

- хранить информацию о системе и данных в разделенных по смыслу блоках кода.

Ещё одним преимуществом AngularJS является возможность разделения клиентской и серверной сторон, что позволяет вести разработку параллельно. Серверный код не смешивается с кодом клиентской стороны, а значит, повышается конечная читабельность.

Наличие готовых решений от фреймворка также благополучно сказалось на судьбе AngularJS по причине того, что он помогает решать довольно разнообразные задачи, используя уже существующие модули. Готовые решения также позволяют программистам сосредоточиться на логике своего приложения, не отвлекаясь на написание дополнительного структурного кода приложения, который как раз и реализуется фрейморком.

В дополнение к этому, отсутствия жесткого каркаса проекта позволяет использовать AngularJS для создания приложений с достаточно разнообразной внутренней структурой или для мобильной разработки.

Другим немаловажным положительным эффектом фреймворка является стандартизация кодирования. AngularJS имеет достаточно низкий порог вхождения по сравнению со многими другими фреймворками, поэтому популярен среди программистов и может использоваться повсеместно, не усложняя чтение кода.

Предлагаемые данным фрейморком «фишки» в виде директив и сервисов являются его главной «изюминкой».

Разработчики AngularJS отошли от традиционной идеи, что язык HTML является врагом веб-программирования. Они решили не бороться с ним, а наоборот, естественным образом расширить этот язык разметки путем ввода дополнительных директив.

Директивы являются одной из ключевых возможностей AngularJS. Они позволяют разработчику описать поведение отдельных элементов и расширить синтаксис языка HTML. Директивы используются для повышения модульности веб-приложения, выделения обособленной функциональности в компоненты и в том числе и для их повторного использования в дальнейшем.

В AngularJS используются также сервисы, которые представляются несколькими встроенными службами (например, $http, которая позволяет создать XMLHttpRequests – объект, для связи с сервером). Сервисы - одноэлементные объекты, которые построены по принципу паттерна одиночки,

т.е. они представлены в системе в единичном экземпляре, доступны из любой части проекта и хранят данные о своем состоянии.

Вместе сервисы и директивы представляют собой мощный инструмент для разработки веб-приложений, который облегчает весь процесс реализации логики приложения.

Что касается недостатков использования фремворка AngularJS, то они вполне прозрачны и заключаются в следующем:

- так как AngularJS является JavaScript-фреймворком, приложения, написанные с его помощью, не являются безопасными. Проверка подлинности на стороне сервера и авторизация обязательно должны быть предусмотрены отдельно.

- если пользователь приложения отключает JavaScript, то он теряет функциональность приложения и большинство информации на страницах.

Выводы. Таким образом, становится понятно, что использование MVC-фреймворков позволяет значительно упростить процесс создания веб-приложений, стандартизировать код, быстро и гибко решить поставленные задачи, а главное – сократить время на разработку программного обеспечения. Поэтому применение таких фреймворков составляет неотъемлемую часть многих постоянно развивающихся как больших, так и малых веб-программ.

Список литературы

1. Малышев, А.В. Проектирование системы автоматизированного перевода для мобильных устройств/ А.А. Родин, В.Г. Сабуров, А.В. Малышев // Программная инженерия: современные тенденции развития и применения: сборник материалов Всероссийской конференции – Курск, 2017. – С. 143-147.

2. Жерденко, К.А. Модификация интерфейса сайта для понятного пользователю отображения семантических URL / К.А. Жерденко, А.В. Малышев // Вестник научных конференций: матер. Междунар. научно-практ. конф. - Тамбов, 2016. – С. 42-44.

3. Жерденко, К.А. Web-code optimizing control system with saving source code readability / К.А. Жерденко, А.В. Малышев // Bridge to science: research works: матер. Междунар. научно-практ. конф. – Сан-Франциско, США, 2017. – С.148-152.

4. Жерденко, К.А. Структура программы для представления алгоритмов управления процессом тестирования с помощью структуры данных /

Т.М. Белова, В.Г. Белов, К.А. Жерденко // Информационные системы и технологии: матер. докладов II международной научно-технич. заочной конференции «ИСТ-2016» - Курск, 2016. – С. 52-54.

5. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Запюгосюун-т. – Курск, 2018. –258с.

6. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

7. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

Жерденко К.А., студент, e-mail: karina.gerdenko@rambler.ru
Малышев А.В., доцент, к.т.н., e-mail: alta76@yandex.ru
ЮЗГУ, г. Курск, Российская Федерация

СОВРЕМЕННЫЙ КОМПЛЕКСНЫЙ ПОДХОД К ТЕСТИРОВАНИЮ ВЕБ-ПРИЛОЖЕНИЙ

В статье приводится характеристика методов тестирования веб-приложений и целесообразность их применения в веб-разработке. Так как в современном мире веб-приложения пользуются наибольшей популярностью среди пользователей из-за своей доступности, они постоянно обрабатываются огромное количество конфиденциальных данных, а поэтому нуждаются в обеспечении полной безопасности соединения между клиентом и сервером. Поэтому важной основой обеспечения безопасности является выполнение правильно проведенного тестирования программного продукта, направленного на выявление ошибок и уязвимостей в веб-коде, для последующего их устранения.

Ключевые слова: *тестирование, веб-приложение, сайт, методы тестирования, защита конфиденциальных данных, функциональное тестирование, тестирование интерфейса, тестирование удобства использования, тестирование производительности, тестирование безопасности.*

MODERN COMPLEX TESTING OF WEB-APPLICATIONS

This article presents the modern web-testing bases. A modern web-application must be secure and invulnerable to provide private user data. That's why the first way to make a web-application secure is full testing, helping to find the bugs and fix it.

Keywords: *testing, a web-application, private user data, functional testing, usability testing, interface testing, load testing, stress testing, security testing.*

В настоящее время повсеместно внедряются веб-технологии, которые, широко применяются в различных сферах человеческой деятельности: в медицину, в систему образования, в систему предоставления государственных услуг и т.д. [1,2,9]. С помощью сети Интернет люди обмениваются конфиденциальными данными, которые не должны быть доступны для посторонних пользователей .

Именно поэтому важно уделять внимание не только разработке программного обеспечения, функционирующего в глобальной Сети, но и его тестированию [3-6].

Процесс проведения тестирования веб-приложений должен соответствовать всем современным требованиям развивающихся веб-технологий и обеспечивать проверку с целью:

- защиты передаваемых пользователями данных;
- работоспособности созданных сценариев использования веб-приложения;
- соединения типа «клиент-сервер»;
- производительности и нагрузки;
- уязвимости системы вцелом.

В связи с тем, что каждое веб-приложение представляет собой программу типа «клиент-сервер», методы, применяемые для тестирования обычных «классических» программ, могут быть использованы для тестирования каждой части веб-приложения в отдельности [4-5].

Далее будут рассмотрены основные подходы к тестированию веб-приложений (рисунок 1):

- функциональное тестирование;
- тестирование пользовательского интерфейса/тестирование верстки;
- тестирование удобства использования;

- тестирование производительности;
- тестирование безопасности.

Рис. 1. Основные подходы к тестированию веб-приложений

Функциональное тестирование представляет собой наиболее продолжительный этап проверки веб-ресурса, обеспечивающий процесс верификации функционирования продукта в соответствии с его начальными спецификациями.

Основные объекты веб-приложения, попадающие под проверку:

- обязательные функции приложения;
- пользовательские формы на интерфейсе (обратная связь, чат, добавление/удаление комментариев, возможность подписки);
- поиск;
- гиперссылки/нерабочие ссылки;
- подгрузка файлов на сервер через клиентскую программу;
- разнообразные дополнительные модули: счётчики, корзина, «фишки» для привлечения клиентов и т.д.;
- вывод и ввод данных;
- контент страниц/дублированный контент.

В связи с этим, процесс функционального тестирования является ничем иным, как процессом нахождения ошибок при помощи имитационного моделирования фактического использования пользователем системы [6-8].

Тестирование пользовательского интерфейса/тестирование верстки представляет собой проверку расположения элементов веб-страниц, соответствие интерфейса веб-страниц согласованным дизайн-макетам и функциональным требованиям системы. Здесь же осуществляется проверка оптимизации изображений и графики, что крайне важно для функционирования веб-приложения.

На этом этапе также происходит проверка валидности кода разметки: соответствует ли код заявленному стандарту, соблюдена ли иерархия объектов, есть ли избыточность элементов [10]. Данные действия обязательны, ведь от них зависит, как будет выглядеть веб-приложение в разных браузерах. Кроссбраузерность – важнейшее техническое условие функционирования веб-приложения, так как каждый из существующих браузеров имеет свои способы обработки веб-кода, а значит, может «по своему усмотрению» интерпретировать ошибки разметки. Это приводит к некорректному отображению форм на используемой платформе.

Тестирование пользовательского интерфейса может проводиться разными методами: как вручную, с помощью, например, специально подобранных пользователей, так и с помощью разнообразных инструментов, взаимодействующих с разметкой веб-программы. Тестирование удобства использования является важным этапом в общем тестировании веб-приложения и проводится именно с реальными пользователями. Оно позволяет получить максимум информации о степени удобства использования веб-ресурса, его понятности и привлекательности в контексте разрабатываемой задачи.

Данное тестирование включает в себя следующие шаги:

- разработка тестовой стратегии, охватывающей весь функционал приложения;
- подбор участников тестирования;
- запуск тестов под контролем экспертов;
- анализ результатов и оптимизация веб-приложения.

Тестирование удобства пользования позволяет оценить программу по следующим пунктам:

- производительность, т.е. оценка количества времени, которое понадобится пользователю для совершения основных действий в приложении;
- правильность, т.е. оценка количества ошибок, сделанных пользователем во время работы с приложением;

• активизация в памяти (привыкание), т.е. насколько последовательность действий в приложении поддается запоминанию человеком и последующему воспроизведению;

• эмоциональная реакция, т.е. как пользователь относится к интерфейсу программы и порядку выполнения действий, раздражает ли что-то пользователя или, наоборот, он рад, что работать с приложением оказалось проще, чем он думал.

С помощью тестирования производительности сайта проводится определение быстродействия и скорости реакции системы, когда она находится под заданной нагрузкой. Здесь можно проследить, как работает приложение при различной скорости интернета и как оно себя поведёт при нормальных и пиковых нагрузках.

Данное тестирование включает в себя следующие виды.

• Нагрузочное тестирование, которое проводится для того, чтобы оценить поведение приложения под заданной ожидаемой нагрузкой (например, количество одновременно работающих пользователей на сайте, совершающих определенные действия по взаимодействию с сервером).

• Тестирование быстродействия, которое обеспечивает проверку скорости загрузки веб-приложения: подгрузку скриптов, изображений, видеофайлов, аудио-файлов, контента большого объема и т.д. Данное тестирование проводится с целью оптимизации процесса загрузки сайта.

Нагрузочное тестирование, в свою очередь, представляет собой:

• стрессовое тестирование, которое помогает определить, насколько система готова к работе при пиковых нагрузках и как поведет себя в случае сбоя;

• объемное тестирование, которое позволяет оценить, насколько программа готова работать при увеличении объемов данных в базе данных.

Тестирование безопасности является завершающим шагом в тестировании веб-приложения. Подразумевается, что, приступая к данному этапу, само приложение соответствует всем требованиям, предъявленным к нему, и не содержит функциональных ошибок.

На данной стадии проверяется следующее:

• доступ к служебным/закрытым страницам неавторизованных пользователей;

• имеют ли все административные страницы защиту от внешнего воздействия;

• защищено ли приложение вцелом от несанкционированного доступа и вредоносных действий;

• защищено ли соединение с сервером;

• защищены ли пользовательские данные от возможности их перехвата;

• прекращение работы открытых сессий после деактивации пользователя в приложении;

• работа SSL-протокола;

• загрузка файлов с ограниченным доступом.

Таким образом, все описанные выше шаги представляют собой цикл тестирования веб-приложения после его создания. Однако такой цикл может быть повторен, например, частично, если на одном из этапов были выявлены ошибки, которые пересекаются с процессом тестирования другого этапа.

Данные методы тестирования веб-приложений позволяют максимально проверить созданный функционал и выявить в нем ошибки.

Список использованных источников

1. Борисовский, С.А. Нейросетевые модели с иерархическим пространством информативных признаков для сегментации плохоструктурированных изображений/ С.А. Борисовский, А.Н. Брежнева, Р.А. Томакова// Биомедицинская радиоэлектроника. 2010. №2. –С.49-53.

2. Малышев, А.В. Проектирование системы автоматизированного перевода для мобильных устройств/ А.А. Родин, В.Г. Сабуров, А.В. Малышев // Программная инженерия: современные тенденции развития и применения: сборник материалов Всероссийской конференции – Курск, 2017. – С. 143-147.

3. Жерденко, К.А. Модификация интерфейса сайта для понятного пользователю отображения семантических URL / К.А. Жерденко, А.В. Малышев // Вестник научных конференций: матер. Междунар. научно-практ. конф. - Тамбов, 2016. – С. 42-44.

4. Жерденко, К.А. Web-code optimizing control system with saving source code readability / К.А. Жерденко, А.В. Малышев // Bridge to science: research works: матер. Междунар. научно-практ. конф. – Сан-Франциско, США, 2017. – С.148-152.

5. Жерденко, К.А. Структура программы для представления алгоритмов управления процессом тестирования с помощью структуры данных /

Т.М. Белова, В.Г. Белов, К.А. Жерденко // Информационные системы и технологии: матер. докладов II международной научно-технич. заочной конференции «ИСТ-2016» - Курск, 2016. – С. 52-54.

6. Белова, Т.М. Генерация заданий по вычислительной (учебной) практике через интернет-сайт / Т.М. Белова, К.Р. Мальцев // Интеллектуальные информационные системы: тенденции, проблемы, перспективы: матер. докладов III региональной заочной научно-практич. конференции «ИИС-2015» - Курск, 2015. – С. 27-29.

7. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

8. Малышев, А.В. Распределенная система для программного управления/ А.В. Малышев, М.В. Медведева, Л.М. Миневич, В.А. Колосков.Патент на изобретение RUS № 2185656 от 16.10.2000.

9. Томакова, Р.А. Гибридные методы и алгоритмы для интеллектуальных систем классификации соложноструктурируемых изображений: автореф.дис. … докт.техн.наук: 05.13.17/Томакова Римма Александровна. – Белгород, 2013. –42с.

10. Аникина Е.И. Информатика: Адаптационный курс: учеб. пособие / Е. И. Аникина, Е. В. Павлова; Юго-Зап. гос. Ун-т. Курск, 2016.- 95 с.

Белова Т.М, доцент, e-mail: tm_belova@mail.ru
Кобелев А.С., студент,
ЮЗГУ, г. Курск, Российская Федерация

ПРИЛОЖЕНИЕ ДЛЯ СОЗДАНИЯ ДИАГРАММ КЛАССОВ

В статье рассматривается программный продукт, предназначенный для создания UML диаграмм классов и генерации шаблона с кодом класса в соответствии с диаграммой.

***Ключевые слова**: объектно-ориентированное программирование, UML, класс, диаграмма класса, интерфейс, графический редактор.*

ANNEX FOR CREATING CLASS DIAGRAMS

The article discusses a software product designed to create UML class diagrams and generate a template with class code in accordance with the diagram.

***Keywords**: object-oriented programming, UML, class, class diagram, interface, graphic editor.*

Наиболее распространенной технологией разработки программных продуктов в настоящее время является объектно-ориентированная технология. Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Идеологически ООП — подход к программированию как к моделированию информационных объектов. Для разработки объектно-ориентированного программного обеспечения широко используется язык моделирования UML [1,2,3,4]. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода. UML позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий, таких как класс, компонент, обобщение, агрегация.

Таблица 1 – Сравнительный анализ программных продуктов и DGCL

Функциональные требования	Программные продукты			
	UMLet	Eclipse	yEd	DGCL
Возможность создания диаграмм классов	✓	✓	✓	✓
Простота интерфейса				✓
Возможность быстрого освоения программного продукта				✓
Возможность сохранения и загрузки проекта	✓	✓	✓	✓
Возможность генерации кода по диаграмме класса		✓		✓

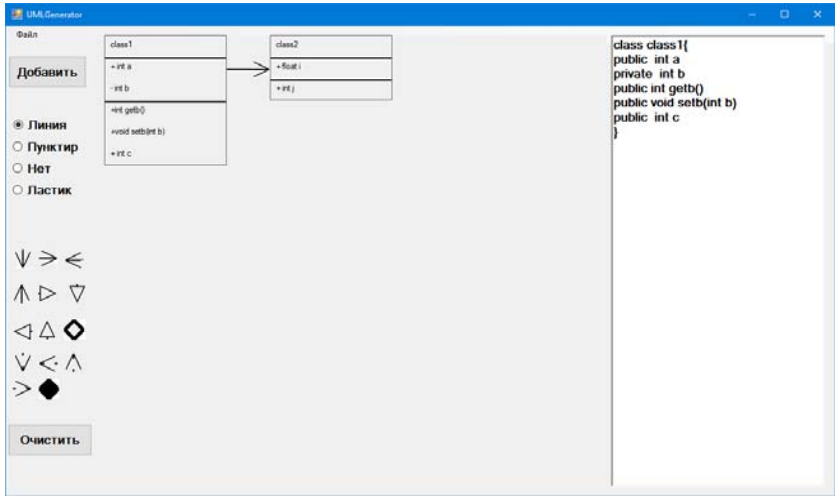


Рис. 1 – Пример работы приложения DGCL

Разработанное приложение для создания диаграмм классов (DGCL) реализовано как графический редактор диаграмм классов и позволяет генерировать код класса, используя диаграмму.

В таблице 1 представлен сравнительный анализ трех существующих программных продуктов и разработанного приложения DGCL.

Из сравнительного анализа существующих программных продуктов и разработанного приложения DGCL, представленного в таблице 1, можно

сделать вывод, что приложение DGCL удовлетворяет всем функциональным требованиям, перечисленным в таблице 1.

Удобный графический интерфейс приложения позволяет разработать UML диаграмму класса и сгенерировать код класса на языке программирования C#. Функциональные возможности программного продукта DGCL можно расширить, обеспечив генерацию кода класса для других современных объектно-ориентированных языков программирования, таких как C++, JAVA. На рисунке 1 представлен пример работы приложения DGCL.

Таким образом, разработанное приложение DGCL удовлетворяет всем функциональным требованиям, предъявляемым к разработке UML диаграмм классов, позволяет быстро получить шаблон кода класса, может использоваться при проектировании объектно-ориентированных программных продуктов на кафедре программной инженерии.

Список использованных источников

1. Пайлон Д. UML 2 для программистов [Текст] / Д. Пайлон, Н. Питмен. – СПб.: Питер, 2012. - 240 с.

2. Атакищев, О.И. Трехуровневая объктно-ориентированная модель организации параллельных асинхронных вычислительных процессов в ГИС [Текст] / О.И. Атакищев, Т.М. Белова, М.В. Белов // Известия Курск. гос. техн. ун-та. - 2004. - №2(13). - С. 67-72.

3. Белова Т.М. Структура программы для представления алгоритмов управления процессом тестирования с помощью структуры данных [Текст]/ Т.М. Белова, В.Г. Белов, К.А. Жерденко // Информационные системы и технологии: материалы докладов II Международной научно-практической заочной конференции «ИСТ -2016». – Курск, ЗАО «Университетская книга», 2016. – С. 52 -54.

4. Белова Т.М. Представление параллельных и асинхронных алгоритмов в виде структур данных [Текст] / Т.М. Белова, Е.С. Кофанова, А.С. Тулупцева // Интеллектуальные информационные системы: тенденции, проблемы, перспективы. Материалы докладов IV международной заочной научно-практической конференции «ИИС-2016» (20 января 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 11 -12.

Корсунский Н.А., студент, e-mail: cor.nick2013@yandex.ru
Карякин Е.С., студент, e-mail:
Алексеев В.А., студент, e-mail: vladislaw.al2015@yandex.ru
ЮЗГУ, г.Курск, Российская Федерация

МЕТОД ПОСТРОЕНИЯ РАЗДЕЛЯЮЩЕЙ ГРАНИЦЫ ВЫБОРОЧНЫХ ОБРАЗОВ, НА ОСНОВЕ ПОТЕНЦИАЛЬНЫХ ФУНКЦИЙ

Рассатривается подход к построению разделяющих границ, основанный на использовании потенциальных функций.

Ключевые слова: *уравнение разделяющей границы, системы распознавания, классификация образов.*

METHOD OF SHARING BOUNDARY OF ELECTIVE IMAGES CONSTRUCTION, ON THE BASIS OF POTENTIAL FUNCTIONS

The approach for the definition of sharing boundaries is considered, based on the usage of potential functions.

Key words*: equation of sharing boundary, recognition system, classification of images.*

При решении задач автоматизированного обучения классификации данных возникает необходимость обработки больших объемов цифровых данных, которые имеют место в связи с применением компьютерных технологий в различных сферах, например, в медицинской диагностике, анализ рисков в банковском деле, фильтрации спама и др. [1-3].

Важным аспектом для успешного функционирования системы распознавания заключается в нахождении решений о принадлежности предъявляемых образов некоторому классу [4-6].

Поэтому актуальной задачей является разработка метода и алгоритма, позволяющего аппроксимировать разделяющие границы на различных наборах информационных данных, включая линейно неразделимые конфигурации классов и получать модели классификации.

Одним из возможных подходов к решению такой задачи является применение решающих функций. Нетрудно показать, что для классов, в состав которых не входят идентичные векторы образов, можно всегда построить

разделяющие границы. Один из этапов при решении этой задачи состоит в обобщении понятия линейной решающей функции, а именно во введении функций вида:

$$d(\vec{x}) = \sum_{i=1}^{N+1} w_i f_i(\vec{x}) \qquad (1)$$

где $N+1$- число членов разложения, $\{f_i(\vec{x})\}$, $i=\overline{1,N}$, - действительные однозначные функции образа $\vec{x}$, $f_{N+1}(\vec{x})=1$. Соотношение (1) представляет бесконечное множество решающих функций, вид которых зависит от выбора функций $\{f_i(\vec{x})\}$, $i=\overline{1,N}$ и количества членов, использованных в разложении.

Продемонстрируем реализацию этого метод с помощью рисунка 1, на котором представлены образы, принадлежащие двум классам.

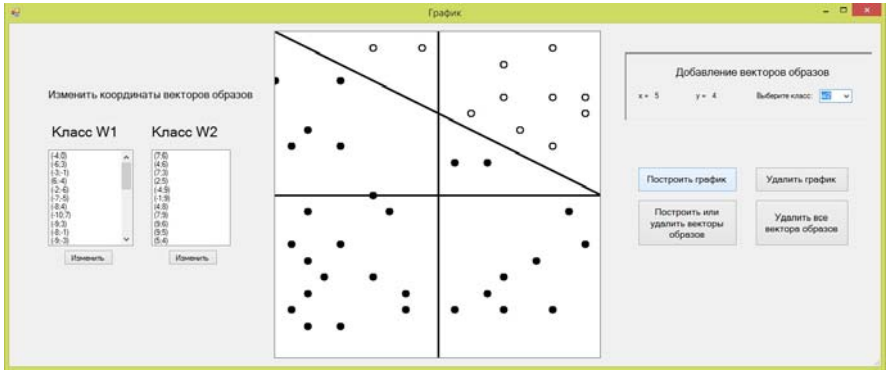


Рис. 1. Построение линейной разделяющей границы

Из рисунка видно, что две совокупности образов возможно разделить прямой вида:

$$d(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3, \qquad (2)$$

где w_i, $i=\overline{1,3}$- коэффициенты, а x_1, x_2 - переменные.

Из рисунка (1) следует, что подстановка в (2) любого образа $\vec{x} \in \omega_1$, принадлежащего классу ω_1 даст положительное значение $d(\vec{x})>0$. При подстановке образа, относящегося к классу ω_2 функция (1) принимает отрицательные значения, $d(\vec{x})<0$. В том случае, когда анализируемый образ лежит на разделяющей границе, имеет место условие неопределенно-

сти $d(\vec{x})=0$ и решение о принадлежности классифицируемого образа одному из классов принимается пользователем.

Успех применения описанной схемы распознавания образов зависит от двух факторов: от вида функции разделяющей границы $d(\vec{x})$ и от практической возможности определения коэффициентов этой функции.

Для проведения классификации выборочных образов различной природы разработан метод, использующий потенциальные функции, позволяющий формировать решающие функции. В основе метода положена гипотеза о существовании эквипотенциальных контуров, которые описываются потенциальной функцией $K(\vec{x},\overrightarrow{x_k})$. С этой целью Выборочные образы, принадлежащие двум классам ω_1,ω_2, представлены точками в двумерном пространстве (в общем случае n - мерном).

Для моделирования и отображения информации используется предположение о том, что кластеры, образованные выборочными образами, из классов ω_1, ω_2, образует «плато», располагаясь на вершинах некоторой группы холмов. Эти два «плато» разделены «долиной», в которой полагают, что потенциал падает до нуля.

В качестве основного принципа, положенного в основу классификации образов, предлагается построение решающих функций на базе потенциальных функций для векторов, представляющих выборочные образы $\overrightarrow{x_k}$, $k=1,2,3,...$, в пространстве образов в виде:

$$K(\vec{x},\overrightarrow{x_k})=\sum_{i=1}^{\infty}\lambda_i^2\varphi_i(\vec{x})\varphi_i(\overrightarrow{x_k}), \qquad (3)$$

где λ_i, $i=1,2,...$ действительные числа, отличные от нуля;

$\varphi_i(\vec{x})$, $i=1,2,...$ ортонормированные функции.

На этапе обучения выборочные образы предъявляются системе, которая последовательно вычисляет значения соответствующих потенциальных функций. Кумулятивный потенциал $K_k(\vec{x})$ на *k-м* шаге итерации определяется совокупностью значений отдельных потенциальных функций, так чтобы при неправильной классификации образа обучающей выборки $\vec{x}_{k+1}$ производилась коррекция значения кумулятивного потенциала. Если же образ классифицирован правильно, то на этом шаге итерации значение кумулятивного потенциала не изменяется.

Для реализации классификаторов была разработана программа построения разделяющих границ (на языке C#), позволяющая добавить, удалить, изменить векторов образов двух классов принадлежности и построить разделяющую границу между двумя классами, а так же стереть ее.

На рисунке 2 представлены выборочные образы, принадлежащие двум классам ω_1,ω_2, которые невозможно классифицировать с помощью линейных разделяющих функций, но разделимых на основе предложенного метода.

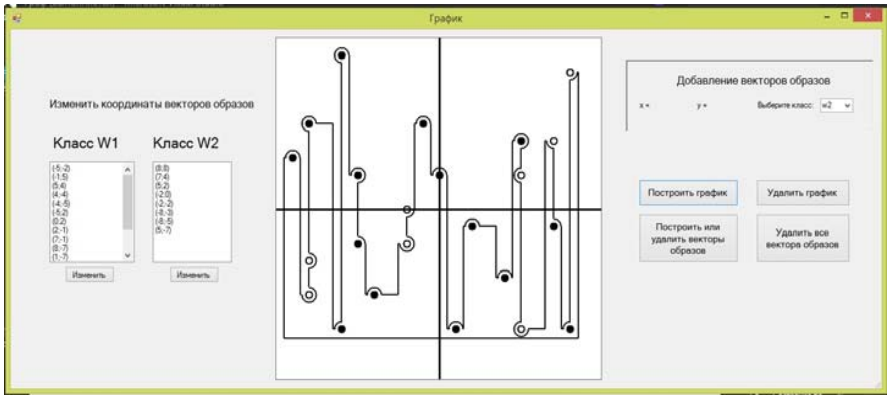


Рис. 2. Построение разделяющей границы на основе потенциальных функций

Вывод. Разработанные на этой математической основе системы распознавания, могут быть предназначены для формирования пространства информативных признаков [7-10], а также позволяют выполнить задачу классификации информативных признаков и строить разделяющие границы в тех случаях, когда представленные для классификации выборочные образы, линейно не разделимы.

Список использованных источников

1. Белобров А.П., Борисовский С.А., Томакова Р.А. Нейросетевые модели морфологических операторов для сегментации изображений медицинских сигналов // Известия ЮФУ. Технические науки. 2010 № 8 (109). С. 28-32.

2. Томакова, Р.А.Построение системы распознавания образов на основе реализации метода потенциальных функций/ Р.А. Томакова, А.Н. Брежне-

ва, Н.А. Корсунский// Мониторинг. Наука и технологии. 2017. №2(31). –С. 46-50.

3. Борисовский С.А., Брежнева А.Н., Томакова Р.А. Нейросетевые модели с иерархическим пространством информативных признаков для сегментации плохоструктурированных изображений // Биомедицинская радиоэлектроника. 2010 № 2С. 49-53.

4. Томакова Р.А., Филист С.А., Жилин В.В., Горбатенко С.А. Структурно-функциональные решения нечетких нейронных сетей для интеллектуальных систем анализа разнотипных признаков // Фундаментальные и прикладные проблемы техники и технологии. 2011 № 1 С. 85-91.

5. Филист, С.А. Метод классификации сложноструктурируемых изображений на основе самоорганизующихся нейронных сетевых структур/ С.А. Филист, Р.А. Томакова, О.В. Шаталова, А.А. Кузьмин//Радиопромышленность. 2016. №4. –С.57-65.

6. Томакова Р.А., Насер А.А., Шаталова О.В. Математическое обеспечение распознавания и классификации сложноструктурируемых биологических объектов // Международный журнал прикладных и фундаментальных исследований. 2012 № 4 С. 48-29.

7. Томакова Р.А., Филист С.А., Руденко В.В. Нечеткая модель интеллектуального морфологического оператора для формирования границ сегментов // Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2011 Т. 17 № 1-1 (96). С. 188-195.

8. Томакова, Р.А. Метод классификации рентгенограмм на основе использования глобальной информации об их структуре/ Р.А. Томакова, М.В. Томаков, И.В. Дураков//Биомедицинская радиоэлектроника. 2016. №9. –С. 45-51.

9. Кореневский, Н.А. Нейронные сети с макрослоями для классификации и прогнозирования патологий сетчатки глаза/А.Н. Кореневский, Р.А. Томакова, С.П. Серегин, А.Ф. Рыбочкин//Медицинская техника.2013. №4.-С.16-18.

10. Дюдин, М.В. Методы и алгоритмы контурного анализа для задач классификации сложноструктурируемых изображений/М.В. Дюдин, А.Д. Поваляев, Е.С. Подвальный, Р.А. Томакова// Вестник Воронежского университета.2014. Т.10. №3-1. –С.54-59.

Кофанова Е.С., студент, e-mail: Sangione-Karin@yandex.ru
ЮЗГУ, г. Курск, Российская Федерация

ПРИМЕНЕНИЕ СИСТЕМЫ РАНЖИРОВАНИЯ В ПРОГРАММЕ ТЕСТИРОВАНИЯ ЗНАНИЙ

В статье рассмотрено применение системы ранжирования в программе для тестирования знаний. Приведены результаты применения данной системы и основные аспекты, а также производится анализ наилучшего ранжирования для более точного выставления оценок.

Ключевые слова: *ранжирование, тестирование, результаты, оценка.*

APPLICATION OF THE RANKING SYSTEM IN THE KNOWLEDGE TEST PROGRAM

The article considers the application of the ranking system in the program for testing knowledge. The results of the application of this system and the main aspects are given, and the best ranking is analyzed for more accurate estimation.

Keywords: *ranking, testing, results, rating.*

Поддержание знаний студентов на нужном уровне является одной из важных и актуальных задач, поскольку грамотный и объективный контроль знаний – залог высокоуровневого и качественного образования [1-3].

Необходимыми средствами для проверки уровня усвоения информации являются средства контроля. Одним из основных направлений улучшения методики усвоенных знаний студентов является использование системы автоматизированного контроля (автоматического тестирования), т.к. традиционные формы контроля недостаточно оперативны, и для их осуществления требуется значительное время [4-5].

В настоящее время тестирование представляет стандартную форму контроля, так как процедура проведения теста, так и оценка знаний единообразна для всех студентов.

Главный принцип, по которому осуществляться проверка в рассматриваемой программе тестирования, является принцип усложненного многогранного отслеживания усвоенной информации: контроль теоретических

и практических знаний учащихся с сложностью, основанной на их успехах во время выполнения заданий на практических занятиях.

В качестве порогового значения, характеризующего удачное прохождение тестирования, на основании экспериментальных исследований было выбрано значение 81%. Если студент дает правильные ответы более 81% вопросов, то тестирование считается успешно завершенным. В противном случае результат не считается удовлетворительным для выставления оценки текущего уровня проверки.

Разработанный программный продукт написан на языке программирования C++ и предназначен для проверки знаний изучающих языки программирования путем тестирования. Данная программа отличается от своих аналогов тем, что позволяет проводить тестирование как с выбором готового варианта ответа, так и с написанием собственного ответа.

Программный продукт содержит следующий уровни тестирования:

1. **Уровень 2 – Оценка «2»**. Поверяет самый низкий уровень знаний. При выборе данного уровня, понижение его невозможно. После удачного прохождения тестирования (более 81%) предоставляется доступ к заданиям уровня 3. Автоматическое выставление оценки не предусмотрено. При неудовлетворительном похождении теста данного уровня происходит оповещение преподавателя и предложение повторного прохождения теста.

2. **Уровень 3 – Оценка «3»**. Поверяет средний уровень знаний. При выборе данного уровня, понижение его при неудачном прохождении (менее 80%) невозможно. После удачного прохождения тестирования (более 81%) предоставляется доступ к заданиям уровня 4. Автоматическое выставление оценки предусмотрено и соответствует оценке «Удовлетворительно».

3. **Уровень 4 – Оценка «4»**. Поверяет уровень знаний выше среднего. При выборе данного уровня, понижение его при неудачном прохождении (менее 80%) допустимо до уровня 3. После удачного прохождения тестирования (более 81%) предоставляется доступ к заданиям уровня 5. Автоматическое выставление оценки предусмотрено и соответствует оценке «Хорошо».

4. **Уровень 5 – Оценка «5»**. Поверяет высокий уровень знаний. При выборе данного уровня, понижение его при неудачном прохождении (менее 80%) допустимо до уровня 3. После удачного прохождения тестирования (более 81%) автоматически выставляется оценка «Отлично».

Интерфейс настройки уровня тестирования и окно результатов прохождения представлены на рисунке 1.

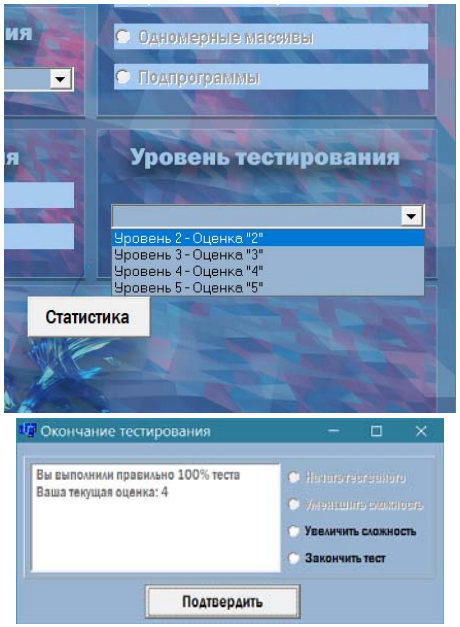


Рис. 1. Настройка уровня тестирования и окно результатов прохождения

Управление данной системой ранжирования происходит за счет применения так называемого алгоритма управления, в основе которого лежит определенная структура данных. В нашем, реализованном случае, представляет мультисписок. Исполнителем алгоритма управления является интерпретатор структуры данных.

Таким образом, разработанная система ранжирования, реализованная в данной программе позволяет наиболее полно оценивать знания учащихся, и выставлять объективно оценки, соответствующие именно их знаниям, исключает возможность воздействия «человеческого фактора».

Список использованных источников

1.Ачасова С. М., Бандман О. Л. Корректность параллельных вычислительных процессов [Текст]/ С.М. Ачасова, О.Л. Бандман. — Новосибирск: Наука, 1990, 253 с.

2. Белова Т.М. Структура программы для представления алгоритмов управления процессом тестирования с помощью структуры данных [Текст]/ Т.М. Белова, В.Г. Белов, К.А. Жерденко // Информационные системы и технологии: материалы докладов II Международной научно-практической заочной конференции «ИСТ -2016». – Курск, ЗАО «Университетская книга», 2016. – С. 52 -54.

3.Белова Т.М. Представление параллельных и асинхронных алгоритмов в виде структур данных [Текст]/ Т.М. Белова, Е.С. Кофанова, А.С. Тулупцева // Интеллектуальные информационные системы: тенденции, проблемы, перспективы. Материалы докладов IV международной заочной научно-практической конференции «ИИС-2016» (20 января 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 11 -12.

4. Аникина Е.И. Информатика: Адаптационный курс: учеб. пособие / Е. И. Аникина, Е. В. Павлова; Юго-Зап. гос. Ун-т. Курск, 2016.- 95 с.

5. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

Белова Т.М., доцент, e-mail: tm_belova@mail.ru
Кузнецов Ю.С., студент,
ЮЗГУ, г. Курск, Российская Федерация

ГЕНЕРАТОР ЗАДАНИЙ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ И УЧЕБНОЙ ПРАКТИКИ

В статье рассматривается программный продукт, предназначенный для выдачи заданий по вариантам для лабораторных работ и учебной практики, его функциональные возможности.

Ключевые слова: *программный продукт, банк заданий, генератор заданий, учебный процесс, интерфейс пользователя.*

TASK GENERATOR FOR LABORATORY WORKS AND EDUCATIONAL PRACTICES

The article considers a software product intended for issuing tasks on options for laboratory work and training practice, its functional capabilities.

Key words*: software product, job bank, task generator, educational process, user interface.*

В учебном процессе на кафедре «Программная инженерия» более десяти лет успешно использовалась программа генерации заданий для учебной практики. Банк заданий по основным разделам программирования содержал более 200 задач. Каждому студенту в соответствии с его порядковым номером в списке группы формировался пакет заданий (обычно четыре задания). Для генерации пакета заданий использовался метод рандомизации.

Однако функциональные возможности данной программы весьма ограничены: вывод заданий осуществляется только на экран и в файл, отсутствуют возможности расширения списков предметов, групп, заданий, печати. Поэтому была поставлена задача разработки программного продукта с улучшенными функциональными возможностями [1,2,3].

На рисунке 1 показаны опции рабочего окна программного продукта для генерации заданий к лабораторным работам по различным дисциплинам и учебной практике.

Для работы со списками студентов необходимо использовать опцию: Файл-Группы. В этом окне можно добавить список новой группы студентов, отредактировать существующий список или удалить список [4,5].

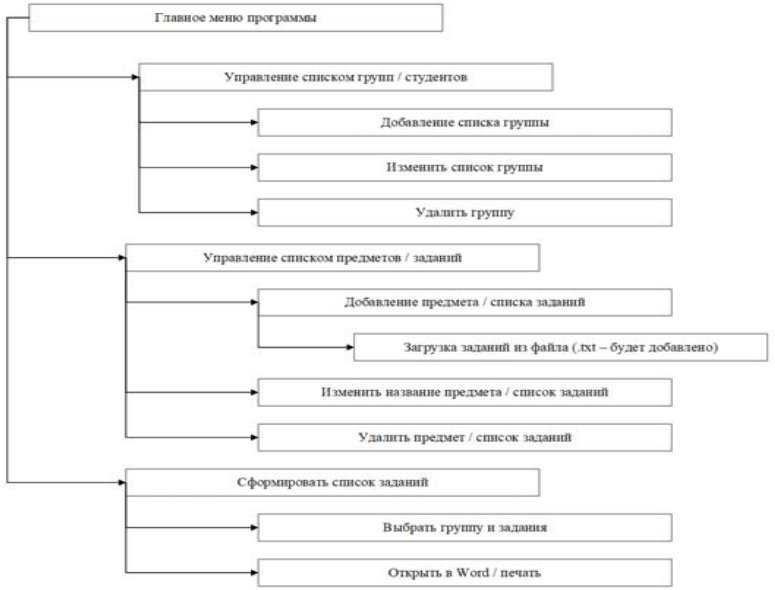


Рис 1. Опции рабочего окна программного продукта для генерации заданий

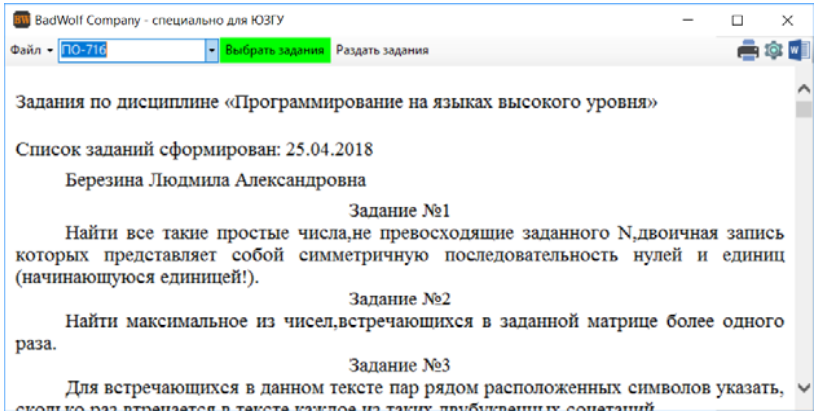


Рис.2. Пример работы генератора заданий

Для работы со списками заданий необходимо перейти к опции: Файл-Задания. В этом окне можно изменить место хранения файлов, добавить

новый предмет/ задание, отредактировать существующие предметы/ задания или удалить предмет/ задание.

Генерация заданий выполняется без повторений заданий, вывод результатов генерации осуществляется на экран, в текстовый редактор WORD, на печать, задания сохраняются в файле.

Пример работы генератора заданий показан на рисунке 2.

Таким образом, разработанный программный продукт обладает: удобным программным интерфейсом; функциональными возможностями, позволяющими осуществлять генерацию заданий без повторений по номеру в списке группы для лабораторных работ по различным дисциплинам и учебной практике, предназначен для использования в учебном процессе для студентов направления подготовки 09.03.04 Программная инженерия.

Список использованных источников

1. Белова Т.М. Генерация заданий по вычислительной (учебной) практике через интернет-сайт [Текст]/ Т.М. Белова, К.Р. Мальцев // Интеллектуальные информационные системы: тенденции, проблемы, перспективы [Текст]: материалы докладов III Региональной заочной научно-практической конференции «ИИС -2015». – Курск, ЗАО «Университетская книга», 2015. – С. 27-29.

2. Белова Т.М. Структура программы для представления алгоритмов управления процессом тестирования с помощью структуры данных [Текст]/ Т.М. Белова, В.Г. Белов, К.А. Жерденко // Информационные системы и технологии: материалы докладов II Международной научно-практической заочной конференции «ИСТ -2016». – Курск, ЗАО «Университетская книга», 2016. – С. 52 -54.

3. Белова Т.М. Представление параллельных и асинхронных алгоритмов в виде структур данных [Текст] / Т.М. Белова, Е.С. Кофанова, А.С. Тулупцева // Интеллектуальные информационные системы: тенденции, проблемы, перспективы. Материалы докладов IV международной заочной научно-практической конференции «ИИС-2016» (20 января 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 11 -12.

4. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. – Курск, 2017. – 194 с.

5. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

Любицкий Н.И., студент, e-mail: nlold@ya.ru
Аникина Е.И., доцент, e-mail: elenaanikina@inbox.ru
ЮЗГУ, г.Курск, Российская Федерация

ПРИМЕНЕНИЕ МЕТОДОВ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ ДЛЯ АНАЛИЗА РАБОТЫ СЛУЖБЫ ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ IT-УСЛУГ

В статье рассмотрено значение службы технической поддержки в ИТ-бизнесе и способы моделирования функционирования службы технической поддержкис помощью методов теории массового обслуживания.

Ключевые слова: техническая поддержка IT-услуг, система массового обслуживания, очередь, время ожидания обслуживания.

APPLICATION OF QUEUING THEORY METHODS FOR THE ANALYSIS OF THE TECHNICAL IT SUPPORT SERVICE

The article considers the importance of the technical support service in the IT business and the analisys of its work with the help of various methods of queuing theory.

***Keywords**: ITsupport service, queuing system, queue, waiting time.*

В современных условиях бизнеса любая компания, которая занимается оказанием IT-услуг, которые направлены на взаимодействие с пользователями, обязана иметь техническую службу поддержки. Техническая поддержка призвана помочь пользователям данного решить конкретные проблемы использования IT-продукта. Таким продуктом может быть как программное обеспечение для конкретной операционной системы или систем, так и веб-приложение [1-3].

Наличие технической поддержки положительно влияет и на пользователей и на разработчиков продукта. Например, в случае затруднения использования программного продукта, пользователь обращается в службу технической поддержки, где ему дают ответы на его вопросы. С одной стороны, поддержка позволяет пользователям получить ответы на вопросы, касающиеся данного продукта, помочь им в этих вопросах и тем самым сохранить лояльность этих пользователей к IT-компании. С другой стороны, такие вопросы от пользователей помогают разработчику быстрее найти и проанализировать разного рода проблемы разрабатываемого продукта [4-6].

Таким образом служба технической поддержки является необходимой для большинства программных продуктов и требует серьезного отношения к её проектированию и разработке. Так как качество обслуживания напрямую влияет на отзывы о продукте и лояльность покупателей/пользователей, то при проектировании нужно учитывать множество внешних факторов, такие как: примерное количество пользователей, время года (например, в праздничные дни может прийти огромное количество заявок в систему, в зависимости от специфики продукта), примерное количество заявок, количество агентов поддержки и т.д. Теоретической основой проектировании подобных систем является теория массового обслуживания, которая занимается изучением систем с элементами случайности, массовостью и процессом обслуживания.

Рассмотрим применение основных понятий теории массового обслуживания к предметной области, связанной с технической службой поддержки. Обслуживаемый объект в теории массового обслуживания называется *заявкой*. Роль заявки, в данном случае, играет заявка от пользователя продукта. Средства, обслуживающие заявки, называются *каналами обслуживания*. Роль каналов в данном случае играют агенты службы поддержки, которые принимают заявки от пользователей. Все заявки попадают в систему массового обслуживания, которая по своей сути является частью данного продукта, отвечающей за техническую службу поддержки.

Системы массового обслуживания классифицируются по разным признакам. Один из них — это количество каналов. Существуют одноканальные (один агент поддержки) и многоканальные (множество агентов поддержки) системы. Далее будут рассматриваться только многоканальные системы массового обслуживания, так как одноканальные абсолютно неадекватны рассматриваемой предметной области [1,7].

Системы массового обслуживания (СМО) классифицируются также по времени пребывания в очереди до начала обслуживания. Существуют системы массового обслуживания с отказами и системы с очередью.

Система с отказами - это система, в которой заявка, поступающая в момент времени, когда все каналы (агенты) заняты, получает отказ в обслуживании и не участвует в дальнейшем процессе обслуживания.

Система с очередью – это система, в которой заявка, поступающая в момент времени, когда все каналы заняты, помещается в очередь на обслуживание. Очередь может быть ограниченной или неограниченной [1].

На основе методов теории массового обслуживания [2,8] нами были проведены расчёты динамики ряда характеристик для трёх моделей СМО: 1) многоканальнаяСМО с отказами в обслуживании, 2) многоканальнаяСМО с ограниченной длиной очереди и 3) многоканальнаяСМО с неограниченной очередью [2,8]. Для каждой из моделей были рассчитаны значения следующих характеристик СМО в зависимости от изменения значений количества агентов и количества поступающих в течение часа заявок на обслуживание:

- интенсивность нагрузки;

$p = \lambda * t_{\text{обс}}$- вероятность, что канал свободен$p_0 = \frac{1}{\Sigma\frac{p^k}{k!}}$;

- вероятность отказа$p_{otk} = \frac{p^n}{n!}p_0$;

- вероятность обслуживания поступающих заявок$p_{\text{обс}} = 1 - p_{отк}$;

- среднее число каналов, занятых обслуживанием$n_з = p * p_{\text{обс}}$;

- абсолютная пропускная способность$A = p_{\text{обс}} * \lambda$;

- среднее время простоя СМО;

$t_{\text{пр}} = p_{\text{отк}} * t_{\text{обс}}$- среднее время простоя канала;

$t_{\text{п.к}} = \frac{t_{\text{обс}} * 1 - p_{\text{отк}}}{p_{\text{отк}}}$- среднее время пребывания заявки в СМО.

$T_{\text{СМО}} = \frac{p_{\text{обс}}}{\mu}$В проведённых нами расчётах среднее время обслуживания клиента $t_{\text{обс}}$ принималось равным 10 минутам и длина очереди принималась равной 5. Сами расчёты в данной статье не приводятся из-за их значительного объёма. Таким образом, можно моделировать и прогнозировать нагрузку многоканальной системы обслуживания на основе данных о количестве агентов поддержки и количестве заявок в час. В качестве основного критерия для сравнения трёх моделей СМО была выбрана оценка процента обслуженных заявок по отношению к количеству поступивших заявок.

На основе полученных в результате расчётов данных можно сделать следующие выводы.

1. Если количество агентов поддержки не превышает 10, количество заявок в час не превышает 35 и их количество можно предугадать, то для

моделирования работы технической службы поддержки лучше всего использовать многоканальную систему массового обслуживания с отказами.

2. Если количество заявок в час более 35, агентов поддержки более 10, то подходит многоканальная система массового обслуживания с ограниченной очередью.

3. Если количеств заявок в час невозможно предугадать, агентов поддержки мало, то лучше всего подходит многоканальная система массового обслуживания с неограниченной очередью.

Список использованных источников

1. Матвеев В.Ф., Ушаков В.Г. Системы массового обслуживания. — М.: МГУ, 1984. — 240 с.

2. Теория массового обслуживания (практикум по решению задач) / САФУ имени М.В. Ломоносова. - Архангельск; САФУ, 2013 — 107 с.

3. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

4. Аникина Е.И. Информатика: Адаптационный курс: учебное пособие / Е. И. Аникина, Е. В. Павлова; Юго-Зап. гос. Ун-т. Курск, 2016.- 95 с.

5. Томакова, Р.А. Метод классификации рентгенограмм на основе использования глобальной информации об их структуре/ Р.А. Томакова, М.В. Томаков, И.В. Дураков//Биомедицинская радиоэлектроника. 2016. №9. –С. 45-51.

6. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/ А.В. Малышев// Известия Юго-Западного государственного университета.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

7. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

8. Харзеева С.Э., Лушникова Е.И. Контекстуальное смысловое моделирование научного дискурса//Русский язык: исторические судьбы и современность . IV Международный конгресс исследователей русского языка: труды и материалы.-М.- 2010. -С. 144-145.

Макашин В.А., студент, e-mail: v.makashin@yandex.ru
Алексеев В.А., студент, e-mail: vladislaw.al2015@yandex.ru
ЮЗГУ, г. Курск, Российская Федерация

ПРОЕКТИРОВАНИЕ БЕЗОПАСНЫХ СМАРТ-КОНТРАКТОВ

В данной статье рассмотрены важные вопросы уязвимости безопасности, возникающие при проектировании и разработки смарт-контрактов на языке solidity.

Ключевые слова: *смарт-контракты, блокчейн, проектирование безопасных смарт контрактов, децентрализованные приложения.*

DESIGNOF SECURE SMART-CONTRACTS

In this article, we examined the main vulnerabilitiesof designing and developing smart contracts on solidity language.

Keywords: *smart-contracts, blockchain, the design of secure smart contracts, Decentralized application.*

Защита информации в компьютерных программах представляет одну из наиболее важных проблем в сфере компьютерных технологий, к решению которой привлечены усилия многих специалистов в сфере информационной безопасности, поскольку данные в компьютерных системах подвержены риску утраты из-за неисправности или уничтожения оборудования, а также риску хищения [1-4].

Безопасность смарт-контрактов является основным приоритетом для децентрализованного приложения, разработанного на основе технологии blockchain [5-8]. Однако погрешности, допущенные в процессе проектирования кода умных контрактов, могут привести к потере ресурсов, а также хищению денежных средств. Основными проблемными узлами при разработке смарт-контрактов являются:

- Выход за предел числовых переменных (Integer Overflow and Underflow)
- Кросс-функциональное состояние гонки (Cross-function Race Conditions)

Следует отметить, что проблема выхода за предел числовых переменных чаще всего встречается при выходе за пределы переменных, при этом как в за нижний предел, так и за верхний предел uint. На рисунке 1 приведен пример фрагмента кода программы с уязвимостью по выходу за предел переменной uint.

```
mapping (address => uint256) public balanceOf;

// INSECURE
function transfer(address _to, uint256 _value) {
    /* Check if sender has balance */
    require(balanceOf[msg.sender] >= _value);
    /* Add and subtract new balances */
    balanceOf[msg.sender] -= _value;
    balanceOf[_to] += _value;
}
```

Рис. 1. Код функции с уязвимостью к переполнению переменной

Для решения данного вопроса необходимо внести исключение при выходе данных за предел числовых переменных, а также быть уверенным в том, что данный контракт может быть использован только ограниченным количеством пользователей, например, если только администратор имеет возможность доступа к данной переменной, то данный контракт может быть безопасен. На рисунке 2 приведен пример фрагмента кода программы, предназначенный для устранения возникающих вопросов уязвимости к переполнению переменной.

```
// SECURE
function transfer(address _to, uint256 _value) {
    /* Check if sender has balance and for overflows */
    require(balanceOf[msg.sender] >= _value && balanceOf[_to] + _value >= balanceOf[_to])

    /* Add and subtract new balances */
    balanceOf[msg.sender] -= _value;
    balanceOf[_to] += _value;
}
```

Рисунок 2 - Код функции исключающий уязвимость выхода за предел переменной

Следует заметить, что проблема кросс-функционального состояния гонки чаще всего возникает при наличии одного состояния у двух функций, что продемонстрировано на рисунке 3.

```
// INSECURE
mapping (address => uint) private userBalances;

function transfer(address to, uint amount) {
    if (userBalances[msg.sender] >= amount) {
       userBalances[to] += amount;
       userBalances[msg.sender] -= amount;
    }
}

function withdrawBalance() public {
    uint amountToWithdraw = userBalances[msg.sender];
    require(msg.sender.call.value(amountToWithdraw)());
    userBalances[msg.sender] = 0;
}
```

Рис. 3. Код программы, использующий одно состояние для двух функций

В этом случае злоумышленник может вызвать функцию transfer (), и когда код будет выполняться во внешнем вызове, тогда злоумышленник может передавать данные, даже если они уже получили вывод.

Вывод. Для того чтобы исключить данную уязвимость и не допустить возможности переполнения переменной пользователю необходимо соблюдать условия: убедиться, что он не вызывает внешнюю функцию, пока не будет выполнена вся внутренняя работа.

Список использованных источников

1. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/ А.В. Малышев// Известия Юго-Западного государственного университета.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

2. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

3. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

4. Аникина Е.И. Информатика: Адаптационный курс: учеб. пособие / Е. И. Аникина, Е. В. Павлова; Юго-Зап. гос. Ун-т. Курск, 2016.- 95 с.

5. Технология распределённого реестра: за рамками блокчейн [Электронный ресурс]. – Режим доступа: http://cryptonyka.com/ files/download/ 31/9bc7b5ca, свободный (Дата доступа 01.10.2017)

6. Decentralized Applications: Harnessing Bitcoin's Blockchain Technology 1-е издание; Siraj Raval, 2016 г. - 150 с.

7. Bitcoin, blockchain, cryptocurrency, cryptology Jeremy Clark; 2016 г. - 499 с.

8. Blockchain: The Simple Guide To Everything You Need To Know 2016 г. -69 с.

Мальцев К.Р., студент, e-mail: SHiFT2501@yandex.ru
Аникина Е.И., доцент, e-mail: elenaanikina@inbox.ru
ЮЗГУ, г. Курск, Российская Федерация

УДАЛЕННОЕ УПРАВЛЕНИЕ РАБОЧИМ СТОЛОМ АВТОМАТИЗИРОВАННОГО РАБОЧЕГО МЕСТАДЛЯ ПРОГРАММЫ СЕТЕВОГО АДМИНИСТРИРОВАНИЯ

В статье рассматриваются этапы разработки программы сетевого администрирования для удалённого управления рабочим столом. Для решения данной задачи была проведена работа для нахождения оптимальных способов получения, обработки и передачи кадров с экрана монитора в браузер пользователя на расстоянии.

***Ключевые слова**: удаленное управление, захват изображения, WebSocket, обработка изображения, передача изображений.*

REMOTE DESKTOP CONTROL OF A WORK STATION FOR THE NETWORK ADMINISTRATION PROGRAM

The article describes the development stages of part of the network administration program for remote desktop management. o solve this problem, work was done to find the best ways to receive, process and transfer frames from the monitor screen to the user's browser at a distance.

Keywords*: remote control, screen capture, WebSocket, image processing, image transfer.*

В настоящее время разработано большое количество программ, предназначенных для удаленного администрирования компьютеров и серверов. Все подобные программы созданы для решения разных задач связанных с администрированием, однако не существует единой программы, которая включала бы в себя все необходимые для администрирования на среднем предприятии средства. Исходя из этого, требуется разработать программу, которая поддерживает различные функции управления удаленными автоматизированными рабочими местами (АРМ). Пример структуры компьютерной сети предприятия, для которой надо решать проблемы администрирования, приведен на рисунке 1.

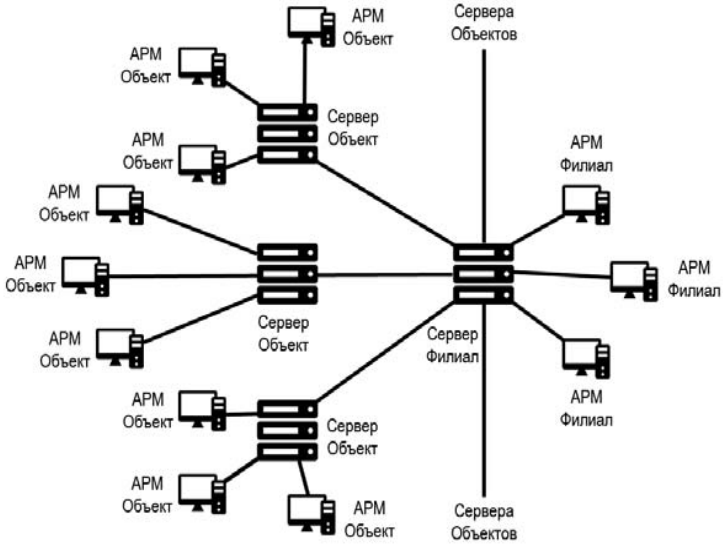


Рис.1. Структура сети предприятия

На АРМ и сервере Объекта будут установлены агенты, разработанные на языке программирования Java, на сервере предприятия будет установлен сервер на «NodeJS», на котором будут «http» и «WebSocket» сервер для передачи данных в браузер на АРМ предприятия. В статье рассмотрена

часть данной программы, предназначенной для сетевого администрирования, отвечающая за удаленный рабочий стол.

Для достижения поставленной цели необходимо было выбрать способ получения изображения с экрана монитора. В начале был рассмотрен класс *Robot* языка Java и его функция *create Screen Capture*. Эта функция создает объект Buffered Image с кодом изображения экрана для дальнейшей обработки и передачи. Однако на среднем АРМ предприятия создание одного снимка занимало примерно 80-100 мс., из-за чего после обработки изображения время обработки возрастало до 150-300 мс., что привело бы к 1-2 кадрам в секунду в браузере пользователя после передачи. Для решения данной проблемы была найдена библиотека *javacv* на языке Java для обработки изображений. С нашей тоски зрения, библиотеки *javacv* имеет ряд недостатков. Во-первых, данная библиотека захватывает экран вместе с курсором. Во-вторых, новые версии библиотеки, начиная с 2016 года, из-за изменений в библиотеке *ffmpeg* не поддерживают старые операционные системы, такие как Windows XP, Windows Server 2003 и т.д., которые могли использоваться на АРМ объектов. Необходимо было найти старую версию программы, поддерживающую старые операционные системы и API. Эта библиотека создает снимок экрана за несколько десятков миллисекунд, в среднем 30 мс на АРМ со средними характеристиками. После обработки время составило 60-150 мс., что дало бы примерно 5-15 кадров в секунду в браузере клиента. Такие цифры устраивали заказчика. Полученную переменную типа *Frame* можно преобразовать при помощи встроенного в библиотеку конвертера в тип *Buffered Image.* Однако, обработка объекта *Buffered Image* занимала много времени, поэтому появилась задача найти способ преобразовать объект *Buffere dImage* в более компактный и удобный для обработки тип данных. В итоге, был выбран массив из байтов, содержащий красный(R), синий (B) и зеленый (G)и каналы. Для перевода использовалось готовое решение[1], которое при помощи нескольких преобразований создавало массив типа байт. Данное решение не отнимает много времени от обработки изображение.

Учитывая, что передается удаленный рабочий стол, можно сделать вывод, что нет смысла передавать каждый кадр заново для того, чтобы перерисовать, например, движение курсора. Можно сохранять предыдущий кадр, сравнивать с текущим и заменять те пиксели, которые не требовали перерисовки пикселями черного цвета(R= 0, G= 0, B= 0). Для того, чтобы не было коллизии черных цветов исходного кадра и новых черных пиксе-

лей, можно в исходном кадре заменять цвет черных пикселей на цвет с увеличенным синим каналом на 1 единицу (R= 0, G= 0, B= 1). Такая замена не заметна и существенно упрощает процесс, особенно учитывая, что чаще всего будет передаваться рабочий стол пользователя.

Не важно, удалены ли совпадающие пиксели текущего и предыдущего кадра, если не применить к ним какой-нибудь алгоритм сжатия. Было проведено сравнение двух вариантов решения данной задачи. Первый вариант предполагал сжимать картинку в формат *Jpeg*, что, возможно, могло бы позволить выбирать качество изображения и достаточно качественно сжимать обработанные кадры с удаленными совпадающими пикселями. Но преобразование обработанного кадра в формат *Jpeg* изображение встроенными библиотеками языка *Java* приводили к повышению времени обработки одного кадра до 150-200 мс, что после передачи не дало достаточно комфортных для пользователя показателей количества кадров в секунду. Поэтому пришлось отказаться от этой идеи. Вторым вариантом решения данной задачи является использования библиотеки сжатия *GZip* . Библиотека *GZip* получает на вход массив типа байт и возвращает массив типа байт, без как-либо ресурсоемких преобразований.

Изображение рабочего стола сжималось до 100-200 кб., а обработанное изображение с перемещением курсора занимало всего 4 кб. Работает «GZip» быстро, по сравнению с первым способом, даже с максимальным сжатием. После захвата изображения с экрана, обработки и сжатия, полное время обработки одного кадра составило 50-100 мс.

Следующим этапом, необходимо было передать кадр с АРМ на промежуточный сервер, с него еще на один сервер и уже затем в браузер на АРМ пользователя. В качестве протокола передачи был выбран WebSocket из-за того, что в качестве промежуточного сервера используется NodeJS. Для улучшения производительности, можно распараллелить процессы на АРМ на 2 потока: захват и обработку кадра, и его сжатие и отправку. Новый кадр обрабатывался в зависимости не от предыдущего, а в зависимости от последнего переданного, таким образом, можно всегда получить актуальную информацию о рабочем столе. Такой подход позволил оптимизировать передачу и обработку кадра до 5-15 кадров в секунду в браузере клиента.

После передачи полученный буфер данных в браузере помещался в массив с типом «Int8Array» и при помощи библиотеки для «GZip» преобразования «pako» преобразовывался в массив с типом «Uint8Array», который содержал описание каждого пикселя в BGR-пространстве. Далее необходимо было обработать данный массив и вывести на экран только изменения. Попиксельная прорисовка такого изображения может занимать длительное время. Решение автора Paul Rouget с Mozilla Hacks [2] помогает решить данную проблему. Он предложил использовать типизированные массивы для этого, так как они работают намного быстрее.

Вывод. После выполнения всех поставленных задач в браузере на АРМ предприятия получен видео поток в 5-15 кадров в секунду, зависящий от динамичности изображения на захватываемом экране. При этом зафиксировано, что в среднем потребляется всего несколько сотен Кбит в сек. Такие показатели, по нашему мнению, являются достаточными для комфортной работы по администрированию удаленного рабочего стола.

Список использованных источников

1. Robert Sutton, Java - get pixel array from image [Электронныйресурс] Режим доступа: https://stackoverflow.com/questions/6524196/java-get-pixel-array-from-image

2. Paul Rouget, Faster Canvas Pixel Manipulation with Typed Arrays [Электронныйресурс] Режим доступа: https://hacks.mozilla.org/2011/12/faster-canvas-pixel-manipulation-with-typed-arrays/

3. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

4. Томакова, Р.А. Структурно-функциональные решения нечетких нейронных сетей для интеллектуальных систем анализа разнотипных признаков/ Р.А. Томакова, С.А. Филист, В.В. Жилин, С.А. Горбатенко//Фундаментальные и прикладные проблемы техники и технологии.2011. №1. –С.85-91.

5. Томакова, Р.А. Гибридные методы и алгоритмы для интеллектуальных систем классификации соложноструктурируемых изображений: автореф.дис. … докт.техн.наук: 05.13.17/Томакова Римма Александровна. – Белгород, 2013. –42с.

6. Томакова, Р.А. Математическое обеспечение распознавания и классификации сложноструктурируемых биологических объектов/ Р.А. Томакова, А.А. Насер, О.В. Шаталова// Международный журнал прикладных и фундаментальных исследований. 2012. №4. 48-29.

7. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/

А.В. Малышев// Известия Юго-Западного государственного университета.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

8. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

9. Харзеева С.Э., Лушникова Е.И. Контекстуальное смысловое моделирование научного дискурса//Русский язык: исторические судьбы и современность . IV Международный конгресс исследователей русского языка: труды и материалы.-М.- 2010. -С. 144-145.

Некрасова Алина Сергеевна, студент, e-mail: schigsv@yandex.ru
Чижова Инна Андреевна, студент, e-mail: chizhova_innochka@mail.ru
ЮЗГУ, г.Курск, Российская Федерация

ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ OPENMP

В статье рассматривается OpenMPстандарт для распараллеливания программ, его формат директив и управление распределением итераций цикла между потоками на языках С.

Ключевые слова:*OpenMP, директивы параллелизма, командные потоки, библиотечные функций, параллельное вычисление,динамический и статический способ.*

PARALLEL PROGRAMMING USING OPENMP TECHNOLOGY

The article discusses the OpenMP standard for parallelizing programs, the format of the directives and control the distribution of loop iterations among the threads in the languages of S.

Keywords: OpenMP, parallelism directives, command streams, library functions, parallel computation,dynamic and static method.

Стандарт для распараллеливания программ OpenMP наиболее широко применяемой в настоящее время для организации параллельных вычислений на многопроцессорных системах с общей памятью. В рамках данной

технологии директивы параллелизма используются для выделения в программе параллельных фрагментов, в которых последовательный исполняемый код может быть разделен на несколько раздельных командных потоков [1-3]. Далее эти потоки могут исполняться на разных процессорах (процессорных ядрах) вычислительной системы. В результате такого подхода программа представляется в виде набора последовательных и параллельных участков программного кода. Подобный принцип организации параллелизма получил наименование "вилочного" (fork-join) или пульсирующего параллелизма [4-7].

Под параллельной программой в рамках OpenMP понимается программа, для которой в специально указываемых при помощи директив местах – параллельных фрагментах – исполняемый программный код может быть разделен на несколько раздельных командных потоков.

Конструктивно в составе технологии OpenMP можно выделить:

- Директивы,
- Библиотеку функций,
- Набор переменных окружения.

Именно в таком порядке и будут рассмотрены возможности технологии OpenMP.

Стандарт предусматривает использование OpenMP для алгоритмических многих языков. Далее описание формата директив OpenMP и все приводимые примеры программ будут представлены на алгоритмическом языке С. В общем виде формат директив OpenMP представлен на рисунке 1:

```
#pragmaomp<имя_директивы> [<параметр>[[,] <пара-
```

Рис.1 Формат директив OpenMP

Начальная часть директивы (#pragmaomp) является фиксированной, вид директивы определяется ее именем (имя_директивы), каждая директива может сопровождаться произвольным количеством параметров (на английском языке для параметров директивы OpenMP используется термин clause).

Для иллюстрации приведем пример директивы: #pragma omp parallel default(shared) \private(beta,pi).

Для балансировки расчетов можно применить статическую или динамическую схемы распределения итераций представлена на рисунке 2.

```
#include <omp.h>
#define CHUNK 100
#define NMAX 1000
main () {
inti, j, sum;
floata[NMAX][NMAX];
<инициализацияданных>
#pragma omp parallel for shared(a) private(i,j,sum) \
schedule(dynamic, CHUNK)
{
for (i=0; i< NMAX; i++) {
sum = 0;
for (j=i; j < NMAX; j++)
sum += a[i][j];
printf ("Сумма элементов строки %d равна %f\n",i,sum);
} /* Завершение параллельного фрагмента */}
```

При разном объеме вычислений в разных итерациях цикла желательно иметь возможность управлять распределением итераций цикла между потоками – в OpenMP это обеспечивается при помощи параметра schedule директивы for. Поле type параметра schedule может принимать следующие значения:

- static – статический способ распределения итераций до начала выполнения цикла. Если поле chunk не указано, то итерации делятся поровну между потоками. При заданном значении chunk итерации цикла делятся на блоки размера chunk и эти блоки распределяются между потоками до начала выполнения цикла;
- dynamic – динамический способ распределения итераций. До началавыполнения цикла потокам выделяются блоки итераций размера;
- guided – управляемый способ распределения итераций. Данный способ близок к предшествующему варианту, отличие состоит только в том, что начальный размер блоков итераций определяется в соответствии с некоторым параметром среды реализации OpenMP, а затем уменьшается экспоненциально следующее значение;

- chunk есть некоторая доля предшествующего значения) при каждом новом выделении блока итераций. При этом получаемый размер блока итераций не должен быть меньше значения chunk;
- runtime – способ распределения итераций, при котором выбор конкретной схемы (из ранее перечисленных) осуществляется в момент начала выполнения программы в соответствии со значением переменной окружения OMP_SCHEDULE.

На данный момент технология OpenMP поддерживается большинством компиляторов языка Си. Несколько хуже дело обстоит с инструментами тестирования параллельных OpenMP программ. Инструменты анализа, проверки и оптимизации параллельных программ хотя и существуют давно, до недавнего времени были мало востребованы при разработке прикладного программного обеспечения. Поэтому они часто являются менее удобными, чем иные инструментальные средства разработки.

Вывод. Наиболее полно процесс разработки параллельных OpenMP программ поддержан в пакете Intel ParallelStudio. Имеется инструмент предварительного анализа кода, для выявления участков кода, которые потенциально можно эффективно распараллелить. Имеется хорошо оптимизирующий компилятор с поддержкой OpenMP. Имеется профилировщик и инструмент динамического анализа для выявления параллельных ошибок.

Список использованных источников

1. Хетагуров, Я.А. Проектирование автоматизированных систем обработки информации и управления / Я.А. Хетагуров. − М.: БИНОМ. Лаборатория знаний. 2015. − 243 с.

2. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

3. Томакова, Р.А. Математическое обеспечение распознавания и классификации сложноструктурируемых биологических объектов/ Р.А. Томакова, Насер А.А., О.В. Шаталова// Международный журнал прикладных и фундаментальных исследований. 2012. −№4. – С.48-49.

4. Quality RTOS &Embedded Software [Электронный ресурс] // Режим доступа – http://www.freertos.org (дата обращения: 25.03.2017).

5. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/ А.В. Малышев// Известия Юго-Западного государственного университе-

та.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

6. Белобров, А.П. Нейросетевые модели морфологических операторов для сегментации изображений медицинских сигналов/ А.П. Белобров, С.А. Борисовский, Р.А. Томакова// Известия ЮФУ. Технические науки. 2010.№8(109). –С.28-32.

7. Филист, С.А. Метод классификации сложноструктурируемых изображений на основе самоорганизующихся нейронных сетевых структур/ С.А. Филист, Р.А. Томакова, О.В. Шаталова, А.А. Кузьмин// Радиопромышленность. 2016. №4. –С.57-65.

Плохих М.Е., студент, e-mail: forsakengod@mail.ru
Аникина Е.И., доцент, e-mail: elenaanikina@inbox.ru
ЮЗГУ, г. Курск, Российская Федерация

ПРОГРАММА TM_ANALISYS ДЛЯ АВТОМАТИЗИРОВАННОГО ПОСТРОЕНИЯ СЛОВАРЯ-МИНИМУМА ПО ИНФОРМАТИКЕ ДЛЯ ИНОСТРАННЫХ СТУДЕНТОВ

В статье рассмотрены проблемы и важность обучения иностранных студентов русскому языку на подготовительном факультете, а также программное обеспечение для составления словаря-минимума, при помощи которого будет происходить обучение лексике русского языка.

***Ключевые слова**:словарь - минимум, лексема, обучение, программное обеспечение, лексика.*

PROGRAM FOR THE AUTOMATED CONSTRUCTION OF A MINIMUM VOCABULARY OFCOMPUTER SCIENCE FOR FOREIGN STUDENTS

The article considers the importance and problems of teaching Russian for foreign students at the preuniversity faculty, as well as software for compiling of a minimum vocabulary in computer science, through which the vocabulary of the Russian language will be taught.

***Keywords:** minimum vocabulary, lexeme, foreign students training, software, lexis.*

В настоящее время рынок образовательных услуг — один из самых интенсивно развивающихся областей, имеющих перспективное будущее, и наравне с культурной индустрией в широком понимании — слагаемое образа страны. Большое количество обучающихся в российских вузах иностранных студентов является значимым индикатором конкурентоспособности российской высшей школы. Помимо очевидных экономических выгод, обучение иностранных студентов в России имеет стратегические преимущества — через подготовку интеллектуальной элиты и распространение русского языка и культуры международное влияние России на мировом рынке повышается [6].

Условием успешного обучения иностранных студентов на русском языке является изучение ими русского языка как иностранного (РКИ)на подготовительном факультете. Одной из важнейших задач изучения РКИ является формирование у студентов минимально необходимого словарного запаса (словаря-минимума), должен включать в себя как общеупотребительные слова, так и термины из изучаемых предметов.

Чтобы обосновать состав учебного словаря-минимумапо изучаемому предмету, необходимо проанализировать текст учебника и выбрать наиболее часто употребляемые существительные, глаголы и др. части речи. Такая работа является достаточно трудоемкой и утомительной, если выполнять ее вручную.

Для автоматизации трудоемкого процесса составлению словаря-минимума нами была разработана программа TM_Analisys, которая выполняет автоматизированный морфологический анализ текстов.

Программа реализована на скриптовом языке Python 3.6. Данный язык был выбран ввиду его простоты и наличия большого числа библиотек, в том числе и для работы с текстами на естественном языке.

Для разработки программы была использована свободно распространяемая библиотека Pymorphy2, выполняющаяморфологический анализ текстов на русском и украинском языках. Pymorphy2 основана на использовании специальным образом обработанных словарей проекта OpenCorpora [4]. OpenCorpora — это проект по созданию размеченного корпуса текстов русского языка.

Библиотека pymorphy2 написана на языке Python (работает в версии языка 2.7 и 3.3+). Возможности данной библиотеки:

•приводить слово к нормальной форме (например, “люди -> человек”, или “гулял -> гулять”).

•ставить слово в нужную форму. Например, ставить слово во множественное число, менять падеж слова и т.д.

•возвращать грамматическую информацию о слове (число, род, падеж, часть речи и т.д.)

При работе используется словарь OpenCorpora; для незнакомых слов строятся гипотезы. Библиотека достаточно быстродействующаяя: в настоящий момент скорость работы - от нескольких тыс. слов/сек до > 100тыс слов/сек (в зависимости от выполняемой операции, интерпретатора и установленных пакетов); потребление памяти - 10...20Мб; полностью поддерживается буква ё [3]

Для выполнения своих функций программе на входе необходим текст в файле формата .txt. Первым этапом в обработке является удаление пунктуации. Удаление пунктуации реализовано с помощью цикла “for”, который построчно проверяет текст на наличие знаков препинания из заранее подготовленного списка. После удаления пунктуации с помощью другого цикла “for”, анализируется каждое слово в обработанном тексте. Анализ производится при помощи метода MorphAnalyzer.parse() из библиотеки pymorphy2. После того, как текст был проанализирован, на основе тегов слов, создается список с выбранной частью речи. Ввиду особенности библиотеки, выбор тега для слова может быть некорректен, поэтому назвать программу автоматической нельзя. Сформированный список слов можно сохранить в памяти компьютера для проверки человеком и дальнейшего использования.

Для обработки были выбраны тексты, используемые в учебном пособии по адаптационному курсу информатики для слушателей иностранного подготовительного отделения [5]. В программу текст загружается в виде файла с расширением .txt. Сначала из текста удаляются знаки пунктуации путем выполнения команды “Удалить пунктуацию”. После удаления знаков пунктуации с помощью команды “Найти глаголы” был сформирован список глаголов в данном тексте.

Опишем кратко функционирование программы на примере составления списка глаголов на основе текстов учебника по информатике. На рисунке 1 представлен внешний вид главного меню программы.

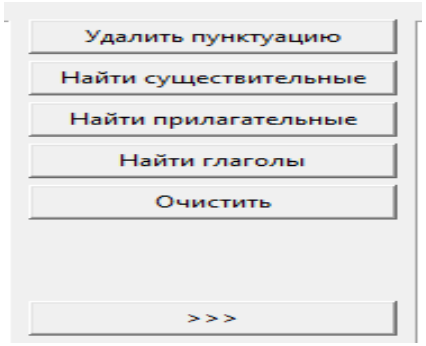


Рис. 1. Главное меню программы

Для сохранения сформированного списка, необходимо нажать кнопку “>>>” и в меню “File” выбрать пункт “Save”. После выбора этого пункта, будет открыт диалог файловой системы, в котором можно выбрать путь для сохранения, указать имя и расширение файла со списком.

В дальнейшем этот перечень глаголов будет включен в словарь-минимум, на основе которого будет производиться обучение иностранных слушателей подготовительного факультета языку информатики.

Список использованных источников

1. Korobov M.: Morphological Analyzer and Generatorf or Russian and Ukrainian Languages//Analysis of Images, Social Networks and Texts, pp. 320-332 (2015).

2. ОриентМикс. Компьютерная обработка текстов: определение и специфика использования [Электрон. ресурс] – Электрон. дан.- Режим доступа: http://pandia.ru/text/80/106/9811.php

3. Морфологический анализатор pymorphy2 [Электрон. ресурс] – Электрон. дан. – Режим доступа: http://pymorphy2. readthedocs.io/ en/latest/ index.html

4. OpenCorpora: открытый корпус русского языка [Электрон. ресурс – Электрон. дан. – Режим доступа:http://opencorpora.org/?page=about

5. Аникина Е.И. Информатика: Адаптационный курс: учеб. пособие / Е. И. Аникина, Е. В. Павлова; Юго-Зап. гос. Ун-т. Курск, 2016.- 95 с.

6. Огнев Д. В., Тулаева Я. И. Обучение иностранных граждан как один из приоритетных показателей конкурентоспособности вуза: состояние, проблемы и перспективы развития [Текст] // Актуальные вопросы эконо-

мических наук: материалы III Междунар. науч. конф. (г. Уфа, июнь 2014 г.). — Уфа: Лето, 2014. — С. 19-22. — URL https://moluch.ru/conf/ econ/ archive/ 95/ 5746/ (дата обращения: 07.02.2018).

7. Харзеева С.Э., Лушникова Е.И. Контекстуальное смысловое моделирование научного дискурса//Русский язык: исторические судьбы и современность . IV Международный конгресс исследователей русского языка: труды и материалы.-М.- 2010. -С. 144-145.

8. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

9. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

Сабуров В.Г., студент, e-mail: v0rub4s@qq.com
Жуков А.А., студент, e-mail: alex.zhukov@temeria.ru
г. Курск, ФГБОУ ВО «Юго-Западный государственный университет»

ВЗАИМОПЕРЕСЕЧЕНИЕ ВЕКТОРНЫХ МОДЕЛЕЙ, ПРЕДСТАВЛЕННЫХ ПРОСТРАНСТВЕННЫМИ ОБЪЕКТАМИ, С ИСПОЛЬЗОВАНИЕМ ТРАНСФОРМИРОВАННЫХ КОРНЕЙ B^+ ДЕРЕВЬЕВ

Аннотация. В статье рассмотрен алгоритм реализации взаимопересечения узлов векторных моделей, которые представлены в виде пространственных объектов с помощью трансформированных корней B^+ деревьев. Предложен и описан концепт новой программной системы.

Ключевые слова: *узлы, корень, объекты, взаимопересечение, алгоритм, реализация, вектор, модель, изображение.*

THE INTERSECTION BETWEEN THE VECTOR MODELS OF SPATIAL OBJECTS, USING THE TRANSFORMED ROOT B+ TREE

Annotation. In the article the algorithm of realization of mutual intersection of nodes of vector models which are presented in the form of spatial objects by means of the transformed roots of B+ trees is considered. The concept of a new software system is proposed and described.

Keywords: *nodes, root, objects, intersection, algorithm, implementation, vector, model, image.*

Векторные изображения представимы в виде совокупности графических примитивов, например, таких как: точка, отрезок, эллипс и т.д., задаваемых математическими формулами, кодирование которых зависит от прикладной среды. Одним из важных достоинств использования векторной графики является то, что файлы, хранящие векторные графические изображения, имеют сравнительно небольшой объем, при этом векторные графические изображения могут быть увеличены или уменьшены без потери качества [1-3].

Векторная модель данных (vector data model) или цифровое представление точечных, линейных и полигональных пространственных объектов в виде набора координатных пар с описанием только геометрии объектов, что соответствует нетопологическоой модели. Векторно-нетопологический формат пространственных данных (vector data format) представляет собой, так называемую, модель «спагетти, в которой каждая точка на карте определяется через ее удаленность от опорной точки и величины угла между направлением на точку из опорной точки и направлением на Север (по часовой стрелке).

Базовым примитивом векторных моделей является точка. Через понятие «точка» определяются все остальные объекты векторной модели.

Безразмерные типы объектов:

- точка – определяет геометрическое местоположение объекта;
- узел – топологический переход или конечная точка, также может определять местоположение объекта.

Одномерные типы объектов:

- линия – одномерный объект;
- линейный сегмент – прямая линия между двумя точками;
- дуга – геометрическое место точек, которые формируют кривую, определенную математической функцией;
- связь – соединение между двумя узлами;
- направленная связь – связь с одним определенным направлением;
- кольцо – последовательность непересекающихся цепочек, строк, связей или замкнутых дуг.

Двумерные типы объектов:

• область – ограниченный непрерывный объект, который может включать или не включать в себя собственную границу;

• внутренняя область – область, которая не включает собственную границу;

• полигон (контур) – двумерный (площадной) объект, внутренняя область которого образована замкнутой последовательностью сегментов в модели «спагетти».

Векторное изображение можно получить различными способами. Наиболее часто используют векторизацию сканированного (растрового) изображения. Векторизация заключается в распознавании на растровом изображении объектов, выделении их, представлении каждого объекта в векторном формате [3-5]. Для автоматической векторизации необходимо иметь изображения высокого качества.

К особенностям векторных моделей можно отнести следующие:

• в векторной модели легко осуществляются некоторые операции с объектами, например, разбивка объекта (речной сети) на участки, замена условных обозначений;

• легко проводятся изменения масштаба, повороты, растягивания и другие операции;

• векторные модели имеют преимущество перед растровыми моделями в точности представления точечных объектов.

Разработан формат пространственных объектов, который представлен в [1] для растровых форм, но он подойдет и для векторных. После определение поля узла, который представляет собой начало отрезка кривой и постепенно заполняющей пространство. При этом значение узла: точка от начала кривой, где первая точка имеет значение 0. А длина отрезка равна степени числа два с максимальным показателем, который кратен адресу начала отрезка. Существует поле, в котором хранится показатель степени числа 2. Отрезок равен 0 только тогда, когда он содержит одну точку.

B^+ деревом называется сбалансированное n-арное дерево поиска порядка t, удовлетворяющее следующим свойствам:

• Каждый узел содержит хотя бы один ключ; ключи в каждом узле упорядочены, корень содержит от 1 до $2t - 1$ключей, любой другой узел содержит от $t - 1$ до $2t - 1$ ключей; листья не являются исключением из этого правила. Здесь t— параметр дерева, не меньший 2 (и обычно принимающий значения от 50 до 2000 в зависимости от размера ключа относительно размера страницы, в свою очередь, определяемого размером содержательной записи);

• У листьев нет потомков; для всех других узлов, содержащих ключи $K_1, \ldots, K_n$, заданный узел содержит $(n+1)$ сыновей.

Построение B+- дерева может требовать перестройки промежуточной структуры, это связано с тем, что количество ключей в каждом узле (кроме корня) должно быть от t до $2t$, где t — степень (или порядок) дерева. При попытке вставить в узел $(2t + 1)$-й ключ возникает необходимость разделить этот узел, в качестве ключа-разделителя сформированных ветвей выступает $(t + 1)$-й ключ, который помещается на соседний ярус дерева. Особым же случаем является разделение корня, так как в этом случае увеличивается число ярусов дерева. Особенностью разделения листа B^+ дерева является то, что он делится на неравные части. При разделении внутреннего узла или корня возникают узлы с равным числом ключей k. Разделение листа может вызвать «цепную реакцию» деления узлов, заканчивающуюся в корне.

Корень B^+ дерева является отправной точкой для всего спектра значений, в котором каждый внутренний узел представляет собой подинтервал.

Например, пусть необходимо взаимопересечение значений ключа k в B^+ дереве. Для этого ищется листовой узел, содержащий значение k. В каждом внутреннем узле нужно выяснить, на какой последующий дочерний узел необходимо следовать, внутренний узел B^+ дерева имеет не более t потомков, где каждый из них представляет собой отдельный подинтервал. Данный метод реализован на языке Phyton. Выбирается соответствующий узел с помощью взаимопересечения в ключевых значениях узла, который представлен на рисунке 1.

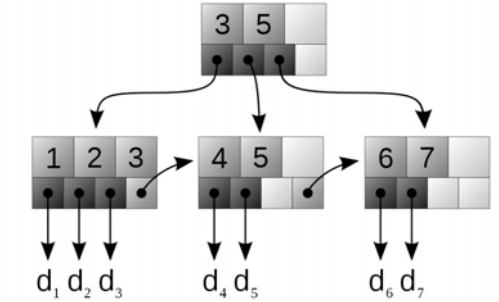


Рис. 1. B^+ дерево с взаимопересечением в узлах

Вывод. Разрабатываемый метод найдет широкие применения, например, при использовании в пространственных баз данных, активно пользующихся географической информационной системой.

Список использованных источников

1. Белов В.Г. Способ кодирования для растровой формы представления пространственных объектов [Текст] / В.Г. Белов, Т.М. Белова // Оптоэлектронные приборы и устройства в системах распознавания образов, обработка изображение и символьной информации. Распознавание 2017. Сборник материалов XIII Международный научно-технический конференции (16– 19 мая 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 63-64.

2. Дюдин, М.В. Методы и алгоритмы контурного анализа для задач классификации сложноструктурируемых изображений/М.В. Дюдин, А.Д. Поваляев, Е.С. Подвальный, Р.А. Томакова// Вестник Воронежского университета. 2014. Т.10. №3-1. –С.54-59.

3. Томакова, Р.А. Метод обработки сложноструктурируемых изображений на основе встроенных функций среды MATLAB/Р.А. Томакова, С.А. Филист//Вестник Забайкальского государственного университета. 2012. №1. –С.3-9.

4. Томакова, Р.А. Гибридные методы и алгоритмы для интеллектуальных систем классификации соложноструктурируемых изображений: автореф.дис. … докт.техн.наук: 05.13.17/Томакова Римма Александровна. – Белгород, 2013. –42с.

5. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/ А.В. Малышев// Известия Юго-Западного государственного университета.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

6. Филист, С.А. Математическая модель системы автоматического регулирования давления в сердечно-сосудистой системе/ С.А. Филист,А.А. Кузьмин, Р.А. Томакова// Системный анализ и управление в биомедицинских системах. 2005.Т4.–№1. – С.50-53.

Севостьянов Д.А., студент, e-mail: d96s@mail.ru
ЮЗГУ, г.Курск, Российская Федерация

СРАВНЕНИЕ МЕТОДОВ ПРЕДСТАВЛЕНИЯ ПРОСТРАНСТВЕННЫХ ОБЪЕКТОВ ДЛЯ ОПЕРАЦИЙ ВСТАВКИ, ОБЪЕДИНЕНИЯ И ПЕРЕСЕЧЕНИЯ

В статье рассмотрены методы представления пространственных объектов. Приведены их затраты по памяти и времени для операции добавления объекта в базу данных. Приведены затраты по времени для операций объединения и пресечения ранее добавленных объектов.

Ключевые слова: *PostgreSQL, PostGIS, представление пространственных объектов, объединение пространственных объектов, пересечение пространственных объектов, q-дерево, кривая Лебега, метод четной адресации.*

COMPARISON OF METHODS FOR REPRESENTATION OF SPATIAL OBJECTS FOR OPERATIONS OF INSERTION, UNION AND INTERSECTION

In this article describes the methods of representation of spatial objects. Memory and time costs for the operation of insertion an object to the database. Time costs for operations of union and intersection previously added objects.

Keywords: *PostgreSQL, PostGIS, representation of spatial objects, union of spatial objects, intersection of spatial objects, q-tree, Lebesgue curve, method even addressing.*

Обработка данных пространственных объектов, представленных в растровой форме, является актуальной задачей [1-8]. Имеется предметная область, на которой заданы объекты. Предметная область задается в виде изображения. Ширина области равна количеству пикселей по горизонтали, а высота количеству пикселей по вертикали. Каждый объект задается изображением с разрешением, как у предметной области. Пиксели фона имеют белый прозрачный цвет. Пиксели с остальными цветами относятся к объекту.

Для хранения данных используется СУБД PostgreSQL.

Первый метод – использование PostGIS. PostGIS – расширение для PostgreSQL, которое позволяет хранить пространственные объекты и выполнять над ними различные операции.

В таблице 1 представлены поля таблицы базы данных для хранения объектов.

Таблица 1 – Поля таблицы базы данных для хранения объектов

Имя	Тип	Описание
nameArea	char(64)	Название области
nameObj	char(64)	Название объекта
geom	geometry	Геометрия объекта

Для добавления объекта необходимо изображение с ним преобразовать в тип geometry. Для перевода изображения с объектом в тип geometry необходимо сначала преобразовать изображение в тип raster, который предназначен для хранения и обработки растровой информации в PostGIS. Затем из raster получить geometry с помощью функции ST_Polygon(raster rast).

Для перевода изображения в тип raster используется утилита PostgreSQL raster2pgsql.exe. Для типа raster нужно задать значение цвета канала, обозначающего отсутствие данных. Это позволит функции ST_Polygon различать пиксели фона и объекта. Делается это с помощью функции ST_SetBandNoDataValue(raster raster, int band, dobule value), где raster – изображение, band – канал (1 – red, 2 – green, 3 - blue), value – значение канала от 0 до 255. Пиксели, с данными значениями каналов будут относится к фону, а остальные к объекту.

Объединение выполняется функцией ST_Union(geometry g1, geometry g2).

Пересечение объектов выполняется функцией ST_Intersection (geometry g1, geometry g2).

Второй метод представляет объект в виде списка отрезков. Отрезок задается адресом начала и адресом конца.

В таблице 2 представлены поля таблицы базы данных для хранения объектов.

Таблица 2 – Поля таблицы базы данных для хранения объектов

Имя	Тип	Описание
nameArea	char(64)	Название области
nameObj	char(64)	Название объекта
start	int	Адрес начала отрезка
end	int	Адрес конца отрезка

Для добавления объекта изображение сканируется построчно.

Для объединения объектов список с отрезками двух объектов сортируется по адресу начала. Два отрезка (s1, e1) и (s2, e2) объединяются в один, если:

$$s1 \le e2 + 1,$$
$$s2 \le e1 + 1;$$

Тогда объединением отрезков будет отрезок (s3, e3), где s3 = s1, e3 = max(e1, e2).

Для пересечения объектов списки с отрезками объектов сортируются по адресу начала. Два отрезка (s1, e1) и (s2, e2) пересекаются, если:

$$s1 \le e2,$$
$$s2 \le e1;$$

Тогда пересечением отрезков будет отрезок (s3, e3), где s3 = max(s1, s2), e3 = min(e1, e2).

В третьем методе объект представляется с помощью линейного квадродерева [2-4]. Для обхода всех пикселей изображения используется кривая Лебега. На рисунке 1 показан обход изображения кривой Лебега.

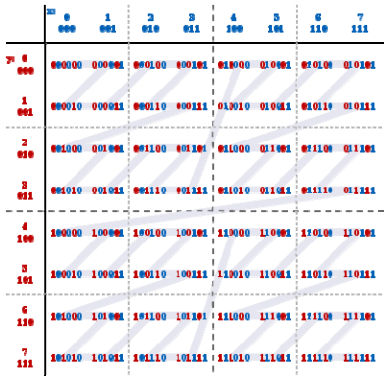


Рис. 1. Обход изображения кривой Лебега

Любой отрезок должен быть длины степени 4. Для начального адреса, представленного кодом Мортона, максимальная степень 4 равна количеству пар нулей на конце. Так, адрес 000000 может иметь длину 1, 4, 16 или 64. А адрес 001000 может иметь длину 1 или 4.

В таблице 3 представлены поля таблицы базы данных для хранения объектов.

Таблица 3 – Поля таблицы базы данных для хранения объектов

Имя	Тип	Описание
nameArea	char(64)	Название области
nameObj	char(64)	Название объекта
start	int	Адрес начала
end	int	Адрес конца

Для объединения объектов список отрезков двух объектов сортируется по возрастанию адреса начала. Два отрезка (s1, e1) и (s2, e2) объединяются в один, если:

$$s1 \le e2,$$
$$s2 \le e1;$$

Тогда объединением отрезков будет отрезок (s3, e3), где s3 = s1, e3 = e2 OR e3.

Отрезки (s1, e1) и (s2, e2) также объединяются в один, если:

$$s2 = e1 + 1,$$

s1 может адресовать e2;

Тогда объединением отрезков будет отрезок (s3, e3), где s3 = s1, e3 = e2.

Для пересечения объектов списки с отрезками объектов сортируются по адресу начала. Два отрезка (s1, e1) и (s2, e2) пересекаются, если:

$$s1 \le e2,$$
$$s2 \le e1;$$

Тогда пересечением отрезков будет отрезок (s3, e3), где s3 = s1 OR s2, e3 = e2 AND e3.

Четвертый метод представления основан на четных адресах. Объект кодируется отрезками, которые могут иметь длину равную степени двойки, т.е. 1, 2, 4, 8 и т.д. Максимальная степень двойки равна количеству нулей в конце у начального адреса в двоичной форме. Так, адрес 1010000 имеет четыре нуля на конце и длина отрезка может быть 1, 2, 4, 8, 16.

Можно заметить, что любой нечетный адрес имеет на конце 1, следовательно, пиксель с нечетным адресом может адресовать отрезок только длины 1.

В таблице 4 представлены поля таблицы базы данных для хранения объектов.

Таблица 4 – Поля таблицы базы данных для хранения объектов

Имя	Тип	Описание
nameArea	char(64)	Название области
nameObj	char(64)	Название объекта
start	int	Адрес начала отрезка
end	int	Адрес конца отрезка

Операции объединения и пересечения выполняются также, как и для третьего метода.

Для сравнения операций над пространственными объектами было сгенерировано 2 набора. Для каждого объекта при генерации пиксель относится к объекту с вероятностью 50%. Особенностью таких объектов является то, что они состоят из множества маленьких областей. На рисунке 2 показан пример объекта для области 64x64.

Рис. 2. Пример объекта для области 64x64

Первый набор содержит 100 объектов для области 4x4.

В таблице 5 приведены результаты измерений для операции добавления объектов для области 4x4.

Таблица 5 – Измерения для добавления объектов

Метод	PostGIS	Отрезки	Степень 4	Степень 2
Время, мс	14822	6179	6284	6322
Память, байт	122880	188416	294912	237568

В таблице 6 приведены результаты измерений для операции объединения объектов для области 4x4.

Таблица 6 – Измерения для объединения объектов

Метод	PostGIS	Отрезки	Степень 4	Степень 2
Время, мс	6638	5959	5963	5861

В таблице 7 приведены результаты измерений для операции пересечения объектов для области 4x4.

Таблица 7 – Измерения для пересечения объектов

Метод	PostGIS	Отрезки	Степень 4	Степень 2
Время, мс	6735	5839	5895	5857

Второй набор содержит 100 объектов для области 64x64.

В таблице 8 приведены результаты измерений для операции добавления объектов для области 64x64.

Таблица 8 – Измерения для добавления объектов

Метод	PostGIS	Отрезки	Степень 4	Степень 2
Время, мс	180254	24315	37350	30897
Память, байт	1368064	35389440	64176128	50855936

В таблице 9 приведены результаты измерений для операции объединения объектов для области 64x64.

Таблица 9 – Измерения для объединения объектов

Метод	PostGIS	Отрезки	Степень 4	Степень 2
Время, мс	121310	7946	11270	10433

В таблице 10 приведены результаты измерений для операции пересечения объектов для области 64x64.

Таблица 10 – Измерения для пересечения объектов

Метод	PostGIS	Отрезки	Степень 4	Степень 2
Время, мс	119584	9019	14779	12936

Таким образом, с объектами, состоящих из множества мелких частей, PostGIS работает медленно. Для области 4x4 три метода, основанные на хранении отрезков, показывают приблизительно одинаковые результаты. Для области 64x64 уже очевиден выигрыш при использовании метода с отрезками. Поэтому, при работе с объектами, состоящими из множества мелких частей, лучше использовать метод представления объектов в виде списка отрезков.

Список использованных источников

1. Атакищев О.И. Отображение графической и атрибутивной информации фрагментов изображения, представленных линейными квадродеревьями, на основе операция реляционной алгебры [Текст] / О.И. Атакищев, А.В. Белов, В.Г. Белов / Наукоемкие технологии. 2012. Т. 13. № 9. С. 34-37.

2. Белов А.В. Представление квадродеревьев бинарными деревьями [Текст] / А.В. Белов, Т.М. Белова / Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. № 1. С. 12-15.

3. Белов А.В. Способы хранения растровых данных на основе квадродеревьев в системах поддержки принятия решений [Текст] / А.В. Белов, Т.М. Белова // Известия Юго-Западного государственного университета. 2012. № 4-2 (43). С. 84-87.

4. Белов В.Г. Представление пространственных объектов отрезками кривых, заполняющих растровое пространство [Текст] / В.Г. Белов, Т.М. Белова // Интеллектуальные информационные системы: тенденции, проблемы, перспективы. Материалы докладов IV международной заочной научно-практической конференции «ИИС-2016» (20 января 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 8 -11.

5. Белов В.Г. Способ кодирования для растровой формы представления пространственных объектов [Текст] / В.Г. Белов, Т.М. Белова // Оптико- электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации. Распознавание 2017. Сборник материалов XIII Международной научно-технической конферен-

ции (16 – 19 мая 2017 г.). – Курск: ЗАО «Университетская книга», 2017. – С. 63- 64.

6. Атакищев, О.И. Трехуровневая объктно-ориентированная модель организации параллельных асинхронных вычислительных процессов в ГИС [Текст] / О.И. Атакищев, Т.М. Белова, М.В. Белов // Известия Курск. гос. техн. ун-та. - 2004. - №2(13). - С. 67-72.

7. Белов В.Г. Определение пересечения пространственных объектов, представленных в растровой форме, с помощью модифицированных B PLUS деревьев [Текст] / В.Г. Белов, Т.М. Белова // Информационные системы и технологии. Сборник материалов III Международной научно-технической конференции. – Курск: ЗАО «Университетская книга», 2017. – С. 56 - 58.

8. Белов В.Г. Определение пересечения пространственных объектов, представленных в растровой форме, с помощью операции естественного соединения [Текст] / В.Г. Белов, Т.М. Белова // Инфотелекоммуникации и космические технологии: состояние, проблемы и пути решения: сборник научных статей по материалам I Всероссийской науч.-практ. конф.: в 2 ч. – Ч. 1 / редкол.: В. Г. Андронов (отв. ред.) [и др.]; Юго-Зап. гос. ун-т. – Курск, 2017. – С. 333 – 335.

Сытченко Д.Ю., студент, e-mail: sit4enko@yandex.ru
ЮЗГУ, г. Курск, Российская Федерация

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Аннотация: *в статье рассмотрен ряд методов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.*

Ключевые слова: *методы классификации, шаблоны, логические методы, деревья решений.*

DATA MINING

Annotation: *in this article a number of methods of detection of previously unknown, non-trivial, practically useful and accessible interpretations of knowledge necessary for decision-making in various spheres of human activity will be considered.*

Key words:*classification methods, patterns, logical methods, decision trees.*

Интеллектуальный анализ данных (ИАД) происходит от английского словосочетания «data mining» используется для: просева информации, извлечения данных, более полным и точным является «обнаружение знаний в базах данных».

Традиционная математическая статистика, долгое время претендовавшая на роль основного инструмента анализа данных, имеет свои существенные недостатки. Главная причина заключается в применении концепции усреднения по выборке, приводящей к операциям над фиктивными величинами. Методы математической статистики оказались полезными, главным образом, для проверки заранее сформулированных гипотез (verification-driven data mining) и для “грубого” разведочного анализа, составляющего основу оперативной аналитической обработки исходной информации.

В основу современной технологии Data Mining положена концепция шаблонов, отражающих фрагменты многоаспектных взаимоотношений в данных. Эти шаблоны представляют собой закономерности, свойственные подвыборкам данных, которые могут быть компактно выражены в понят-

ной исследователю форме. Поиск шаблонов производится методами, не ограниченными рамками априорных предположений о структуре выборке и виде распределений значений анализируемых показателей [1-3].

Перед использованием алгоритмов необходимо произвести подготовку набора анализируемых данных [4-6]. Применение метода интеллектуального анализа данных (ИАД) позволяет обнаружить только присутствующие в данных закономерности, при этом исходные данные, с одной стороны, должны иметь достаточный объём, чтобы эти закономерности в них присутствовали, а с другой — быть достаточно компактными, чтобы анализ занял приемлемое время для исследования [7-10]. Чаще всего в качестве исходных данных выступают хранилища или витрины данных.

В этом случае отфильтрованные данные сводятся к наборам, который в соответствии с гипотезами о том, какие признаки сырых данных имеют высокую прогнозную силу в расчете на требуемую вычислительную мощность для обработки. Например, черно-белое изображение лица размером 100×100 пикселей содержит 10 тыс. бит сырых данных. Они могут быть преобразованы в вектор признаков путём обнаружения на изображении глаз и рта. В результате проведенных преобразований происходит уменьшение объёма данных с 10 тыс. бит до списка кодов положения, значительно уменьшая объём анализируемых данных, а значит и время анализа.

Все методы Data Mining подразделяются на две большие группы, в зависимости от принципа работы с исходными обучающими данными.

1. Непосредственное использование данных, или сохранение данных.

В этом случае исходные данные хранятся в явном детализированном виде и непосредственно используются на стадиях прогностического моделирования или анализа исключений. Проблема этой группы методов - при их использовании могут возникнуть сложности анализа сверхбольших баз данных. Методы этой группы: кластерный анализ, метод ближайшего соседа, метод k-ближайшего соседа, рассуждение по аналогии.

2. Выявление и использование формализованных закономерностей, или дистилляция шаблонов.

При технологии дистилляции шаблонов один образец информации извлекается из исходных данных и преобразуется в некие формальные конструкции. Методы этой группы: логические методы, методы визуализации, методы кросс-табуляции.

Логические методы включают: нечеткие запросы и анализы; символьные правила; деревья решений; генетические алгоритмы. Методы этой

группы являются наиболее интерпретируемыми - они оформляют найденные закономерности в достаточно прозрачном виде с точки зрения пользователя. Полученные правила могут включать непрерывные и дискретные переменные.

Вывод. Методы data mining могут быть широко использованы в инновационных проектах в области проектирования и применения более совершенных информационных технологий как для работы с большими данными, так и для обработки сравнительно малых объемов данных. В качестве критерия достаточного количества данных рассматривается как область исследования, так и применяемый алгоритм анализа.

Список использованных источников

1. Томакова, Р.А. Структурно-функциональные решения нечетких нейронных сетей для интеллектуальных систем анализа разнотипных признаков/ Р.А. Томакова, С.А. Филист, В.В. Жилин, С.А. Горбатенко//Фундаментальные и прикладные проблемы техники и технологии.2011. №1. –С.85-91.

2. Томакова, Р.А. Математическое обеспечение распознавания и классификации сложноструктурируемых биологических объектов/ Р.А. Томакова, Насер А.А., О.В. Шаталова// Международный журнал прикладных и фундаментальных исследований. 2012. −№4. – С.48-49.

3. Quality RTOS & Embedded Software [Электронный ресурс] // Режим доступа – http://www.freertos.org (дата обращения: 25.03.2017).

4. Томакова, Р.А. Метод обработки сложноструктурируемых изображений на основе встроенных функций среды MATLAB/Р.А. Томакова, С.А. Филист//Вестник Забайкальского государственного университета. 2012. №1. –С.3-9.

5. Борисовский, С.А. Нейросетевые модели с иерархическим пространством информативных признаков для сегментации плохоструктурированных изображений/ С.А. Борисовский, А.Н. Брежнева, Р.А. Томакова// Биомедицинская радиоэлектроника. 2010. №2. –С.49-53.

6. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

7. Белобров, А.П. Нейросетевые модели морфологических операторов для сегментации изображений медицинских сигналов/ А.П. Белобров, С.А.

Борисовский, Р.А. Томакова//Известия ЮФУ. Технические науки. 2010.№8(109). –С.28-32.

8. Томакова, Р.А. Нейросетевые модели принятия решений для диагностики заболеваний легких на основе анализа флюорограмм грудной клетки/ Р.А. Томакова, М.В. Дюдин, М.В. Томаков//Биомедицинская радиоэлектроника. 2014.№9. –С.12-15.

9. Томакова, Р.А. Гибридные технологии выделения медленных волн из квазипериодических сигналов/ Р.А. Томакова, А.М. Ефремов, С.А. Филист, О.В. Шаталова// Известия Юго-Западного государственного университета. 2011. №1(34). – С.66-73.

10. Томакова, Р.А. Гибридные методы и алгоритмы для интеллектуальных систем классификации соложноструктурируемых изображений: автореф.дис. … докт.техн.наук: 05.13.17/Томакова Римма Александровна. – Белгород, 2013. –42с.

Чижова И.А., студент, e-mail: chizhova_innochka@mail.ru
Некрасова А.С., студент, e-mail: schigsv@yandex.ru
ЮЗГУ, г.Курск, Российская Федерация

ОСОБЕННОСТИ РЕАЛИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА CUDA

В статье рассмотрена платформа параллельных вычислений CUDA, технологи её реализации, приведены основные характеристики и возможный шаблон для решения задач.

Ключевые слова: *CUDA, платформа, архитектура, программно-аппаратная архитектура, параллельное вычисление.*

TOPICALITY OF PARALLEL COMPUTINGS ON CUDA

The article considers the CUDA parallel computing platform, its technology, characteristics and problem solving pattern.

Keywords: *CUDA, platform, architecture, software and hardware architecture, parallel computing.*

В наше время наиболее актуальна тема принятия решений, для поддержки которых необходимо работать с большим количеством информации, уметь правильно классифицировать ее по необходимым критериям, для последующего анализа и вывода результата.

Устройства, благодаря которым из персональных компьютеров получались маленькие суперкомпьютеры, известны давно. В 80-х годах на рынке предлагались транспьютеры, которые вставлялись в слоты расширения ISA. Первое время они имели впечатлительную производительность в соответствующих задачах, но затем рост быстродействия универсальных процессоров ускорился, они усилили свои позиции в параллельных вычислениях, и смысла в транспьютерах не осталось [1-3].

В последнее время параллельные вычисления перешли к массовому рынку, связанному с играми в 3D-графике. Универсальные устройства с многоядерными процессорами для параллельных векторных вычислений достигают высокой пиковой производительности, которая универсальным процессорам не под силу. Правда в том, что максимальная скорость достигается лишь в ряде удобных задач и имеет некоторые ограничения, но такие устройства уже начали довольно широко применять в сферах, для которых они изначально и не предназначались. Например, процессор Cell, который применяется в игровой приставке SonyPlayStation 3, а также и все современные видеокарты от лидеров рынка — компаний NVIDIA и AMD [4-5].

Компания NVIDIA выпустила платформу CUDA — C-подобный язык программирования со своим компилятором и библиотеками для вычислений на GPU. CUDA – программно-аппаратная архитектура параллельных вычислений, которая позволяет существенно увеличить вычислительную производительность благодаря использованию графических процессоров фирмы NVIDIA [6-8].

Архитектура CUDA предоставляет разработчику широкие возможности для организации доступа к набору инструкций графического ускорителя и позволяет управлять его памятью, по своему усмотрению.

Технология CUDA строится на следующем принципе работы: GPU или по-другому device — устройство является массивно-параллельнымсопроцессором для CPU или host, где выполняется последовательный код [9-10]. Параллельный код, применяемый для параллельных вычислений производится на GPU несколько раз на нескольких параллельных нитях, а функции, которые работают параллельно на всех нитях гра-

фического процессора являются ядрами kernel. Такие функции являются расширенными функциями языка Си.

Основные характеристики CUDA:

– унифицированное программно-аппаратное решение для параллельных вычислений на видеочипах Nvidia;

– большой набор поддерживаемых решений, от мобильных до мультичиповых

– стандартный язык программирования Си;

– стандартные библиотеки численного анализа FFT (быстрое преобразование Фурье) и BLAS (линейная алгебра);

– оптимизированный обмен данными между CPU и GPU;

– взаимодействие с графическими API OpenGL и DirectX;

– поддержка 32-битных и 64-битных операционных систем: Windows XP, WindowsVista, Linux и MacOS X;

– возможность разработки на низком уровне.

Рассмотрим типичный, но не обязательный шаблон решения задач. Для начала задача разбивается на подзадачи [7,9,10],. Входные данные делятся на блоки, которые вмещаются в разделяемую память. Далее каждый блок обрабатывается блоком потоков, который подгружается в разделяемую память из глобальной. Затем над данными в разделяемой памяти проводятся соответствующие вычисления. И в конце результаты копируются из разделяемой памяти обратно в глобальную.

Вывод. Согласно проведенному анализу, можно утверждать, что в перспективе реализации многочисленных вычислений будут производиться на основе применения параллельных алгоритмов. Так как обработку исходного большого блока данных можно разбить на малые порции, не превышающие объема разделяемой памяти, то скорость обработки может быть существенно увеличена, за счет того, что разделяемая память является самым быстрым средством обмена данными между процессами.

Вполне вероятно, что в силу широкого распространения видеокарт в мире, развитие параллельных вычислений на GPU сильно повлияет на индустрию высокопроизводительных вычислений. Представленная компанией NVIDIA программно-аппаратная архитектура для расчётов на видеочипах CUDA хорошо подходит для решения широкого круга задач с высоким параллелизмом.

Список использованных источников

1. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб: БХВ-Петербург, 2015. — 608 с. — ISBN 5-94157-160-7;

2. Оленев Н. Н. Основы параллельного программирования в системе MPI. — М.: ВЦ РАН, 2014. — 80 с. — ISBN 5201098320;

3. NVIDIA CUDA – неграфические вычисления на графических процессорах [Электронный ресурс] - режим доступа: www. ixbt. com/video3/cuda-1. Shtml;

4. Сандерс Д. Технология CUDA в примерах. Введение в программирование графических процессов / Д. Сандерс, Э. Кэндрот-М.: ДМК Пресс, 2011. -232 с.

5. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

6. Томакова, Р.А. Методы и алгоритмы теории принятия решений: учебное пособие /Р.А. Томакова, В.В. Апальков; Юго-Зап. гос. ун-т. – Курск, 2015.-164с.

7. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

8. Аникина Е.И. Информатика: Адаптационный курс: учеб. пособие / Е. И. Аникина, Е. В. Павлова; Юго-Зап. гос. Ун-т. Курск, 2016.- 95 с.

9. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

10. Кореневский, Н.А. Нейронные сети с макрослоями для классификации и прогнозирования патологий сетчатки глаза/А.Н. Кореневский, Р.А. Томакова, С.П. Серегин, А.Ф. Рыбочкин//Медицинская техника.2013. №4.- С.16-18.

Шепелева Ж.А., студент, e-mail:schepelewa.janna@yandex.ru
Аболмасова Е.С., студент, e-mail:evgeniya.abolmasova@gmail.com
Емельянова Е.С., студент, , e-mail: eemelyanova12@yandex.ru
ЮЗГУ, г. Курск, Российская Федерация

АНАЛИЗ ФУНКЦИОНИРОВАНИЯ ТЕХНОЛОГИИNOSQL, РЕАЛИЗОВАННОЙ НА ПРИМЕРЕ MONGODB

В статье рассмотрены методология разработки нереляционных баз данных, а также основные преимуществаMongoDB перед известными реляционными базами данных.

Ключевые слова: *MongoDB, SQL, база данных(БД), система управления базами данных(СУБД), ключ.*

FUNCTIONAL ANALYSIS OF THE NOSQL APPROACH THAT IS IMPLEMENTED ON EXAMPLE OF THE MONGODB

The article describes the methodology of development of non-relational databases, as well as the main advantages of MongoDB over known relational databases.

Keywords: MongoDB, SQL, database(DB), database management system(DBMS), key.

MongoDB представляет сосбой документо-ориентированную систему управления базами данных (СУБД) с открытым исходным кодом. Методология MongoDB реализует новый подход к построению баз данных(БД), в которых нет таблиц, схем, запросов SQL, внешних ключей и многих функций, которые присущи объектно-реляционным БД. В отличие от реляционных БД предлагает документо-ориентированную модель данных, благодаря чему позволяет повысить скорость, обладает лучшей масштабируемостью, ее легче использовать [1,4,5].

MongoDB–СУБД с открытым исходным кодом, классифицируется как NoSQL и использует JSON(JavaScriptObjectNotation)-подобные документы и схему базы данных. Следует отметить, что JSON эффективно описывает сложные по структуре данные, и для хранения в MongoDB применяется формат, который называется BSON (БиСон) или сокращение от binary JSON.

MongoDB написана на C++, поэтому ее легко портировать на самые разные платформы,она может быть развернута на платформах Windows, Linux, MacOS, Solaris. При этом запросы могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям.

Использование рассмотренной СУБД позволяет работать с набором реплик, который состоит из двух и более копий данных. Каждый экземпляр набора реплик может в любой момент выступать в роли основной или вспомогательной реплики. Все операции записи и чтения по умолчанию осуществляются с основной репликой. Вспомогательные реплики поддерживают в актуальном состоянии копии данных. В случае, когда основная реплика дает сбой, набор реплик проводит выбор, который из реплик должен стать основным. Второстепенные реплики могут дополнительно являться источником для операций чтения.

MongoDB масштабируется горизонтально, используя шардинг. При этом пользователь выбирает ключ шарда, который определяет как данные в коллекции будут распределены. Затем данные разделятся на диапазоны (в зависимости от ключа шарда) и распределятся по шардам [2,6,8]..

Следует отметить, что MongoDB может быть использован в качестве файлового хранилища с балансировкой нагрузки и репликацией данных.JavaScript может использоваться в запросах, функциях аггрегации и отправлен базе для исполнения [2,4,7]

Если реляционные базы данных хранят строки, то MongoDB хранит документы. В отличие от строк документы могут хранить сложную по структуре информацию. Документ можно представить как хранилище ключей и значений. Ключ представляет простую метку, с которым ассоциировано определенный объем данных.

Однако при всех различиях есть одна особенность, которая сближает MongoDB и реляционные базы данных. В реляционных СУБД встречается такое понятие как первичный ключ. Это понятие описывает некий столбец, который имеет уникальные значения. В MongoDB для каждого документа имеется уникальный идентификатор, который называется id. И если явным образом не указать его значение, то MongoDB автоматически сгенерирует для него значение.

Если в традиционном мире SQL имеются таблицы, то в MongoDB формируются коллекции. И если в реляционных БД таблицы хранят однотипные жестко структурированные объекты, то в коллекции могут содер-

жать самые разные объекты, имеющие различную структуру и различный набор свойств.

Отсутствие жесткой схемы базы данных, и в связи с этим, потребности при малейшем изменении концепции хранения данных пересоздавать эту схему значительно облегчают работу с БДMongoDB и дальнейшим их масштабированием. Кроме того, экономится время разработчиков. Им больше не надо думать о пересоздании базы данных и тратить время на построение сложных запросов .

Одной из проблем при работе с любыми системами баз данных является сохранение данных большого размера. Можно сохранять данные в файлах, используя различные языки программирования. Некоторые СУБД предлагают специальные типы данных для хранения бинарных данных в БД (например, BLOB в MySQL).

В отличие от реляционных СУБД MongoDB позволяет сохранять различные документы с неодинаковым набором данных, однако при этом размер документа ограничивается 16 мб. Однако MongoDB предлагает специальную технологию GridFS, которая позволяет хранить данные по размеру значительно больше 16 мб [3,4,5]..

Важно отметить, что система GridFS состоит из двух коллекций. В первой коллекции, называемойfiles, хранятся имена файлов, а также их метаданные, например, размер. Во второй коллекции, которая называется chunks,хранятся данные файлов в виде небольших сегментов, обычно сегментами по 256 кб.

Вывод: Таким образом, использование MongoDB позволяет значительно упростить работу с БД, благодаря тому, что в ней реализуется совершенно новый подход к созданию БД, который может быть использован при реализации программных продуктов различного назначения.

Список использованных источников

1. Бэнкер, К. MongoDBв действии / К.Бэнкер. – ДМК Пресс.2017. – 394с.

2. Садаладж П. NoSQL. Новая методология разработки нереляционных баз данных/ П. Садаладж, М. Фаулер. – Вильямс.2017. – 192 с.

3. Редмонд Э. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL/ Э. Редмонд, Д. Уилсон. – ДМК Пресс.2018. – 384 с.

4. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

5. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

6. Малышев, А.В. Организация обменных взаимодействий в мультипроцессоре с использованием данных о текущем состоянии его элементов/ А.В. Малышев// Известия Юго-Западного государственного университета.Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. №1. –С.198-201.

7. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

8. Борисовский, С.А. Нейросетевые модели с иерархическим пространством информативных признаков для сегментации плохоструктурированных изображений/ С.А. Борисовский, А.Н. Брежнева, Р.А. Томакова// Биомедицинская радиоэлектроника. 2010. №2. –С.49-53.

Туев Н.В., студент, e-mail: nituev7@gmail.com
Томакова Р.А., профессор, д.т.н., e-mail: rtomakova@mail.ru
ЮЗГУ, г. Курск, Российская Федерация

ПРИМЕНЕНИЯ АГЕНТНОГО МОДЕЛИРОВАНИЯ В WIKI-СИСТЕМАХ

В статье рассмотрены основные принципы агентного моделирования, которыми следует руководствоваться при использовании имитационного моделирования.

Ключевые слова: *агентное моделирование, имитационное моделирование, агенты, клеточные автоматы, wikiномика, wiki-системы.*

THE APPLICATION OF AGENT-BASED MODELING IN WIKI SYSTEMS

The article considers the basic principles of agent modeling, which should be used when using simulation.

Key words: *agent modeling, imitation modeling, agents, cellular automata, wikionomics, wiki-systems.*

В настоящее время во всем мире наблюдается повышение интереса к возможностям коллективной сетевой деятельности, к расширению экспертного сообщества, включению в него новых заинтересованных участников, обладающих своим взглядом на какую-либо проблему, требующие решения [1-3].

Развитие социальных сервисов привело к возникновению феноменов, которые называют по-разному: мудрость толпы, мобилизация ресурсов людей посредством информационных технологий с целью решения поставленных задач, wikiномика, общественная поддержка, паутина соучастия. В основании этих феноменов лежит возможность привлечения широких масс к непосредственному участию в коллективном творчестве и принятии решений [4-6]. Спектр возможных направлений участников такого творчества охватывает как сравнительно простые действия, например, сбор и повторное использование существующих знаний иконтент-объектов (коллекций медийных материалов, ссылок и т.п.), так и гораздо более

сложные задачи по созданию новых коллективных документов, книг, стандартов и т.д.

Обозначенная проблема, в ряде случаев, решается с помощью агентного моделирования (англ. agent-based model (ABM)), представляюшего собой метод имитационного моделирования, который исследует поведение децентрализованных агентов и определяет условия функционирования всей системы в целом [7-8]. Отличительная особенность агентного моделирования, например от системной динамики, заключается в том, что аналитик определяет поведение агентов на каждом индивидуальном уровне, а глобальное поведение системы в целом характеризуется как результат деятельности множества различных агентов (т.е. моделирование выполняется «снизу вверх»).

Существенным моментом, определяющим практическую значимость агентного моделирования, является его эффективность в условиях различных ограничений, что позволяет использование теоретических основ клеточных автоматов, элементов теории игр, сложных систем, мультиагентных систем и эволюционного программирования, статистических методов [4-6].

Среди многообразия способов организации совместной деятельности наибольший интерес вызывают wiki-системы, в которых авторы работают над коллективными wiki-страницами. Следует отметить, что гипертекст и гипертекстовые веб-технологии изначально предполагали групповую работу с документами. Wiki представляет простую и радикальную модель коллективного гипертекста, в которой возможность создания и редактирования любой записи предоставлена каждому из членов сетевого сообщества [7-10].

В рамках данной работы wiki рассматривается как экологическая система, состоящая из сообщества живых организмов, среды их обитания – множество объектов, системы связей, осуществляющей обмен веществом и энергией между ними. Логическая схема wiki-систем представлена на рисунке 1.

Рис. 1. Логическая схема wiki-систем

В качестве основного принципа при построении модели wiki-системы необходимо придерживаться нескольких простых правил:

- в системе существуют три типа агентов – участники, страницы и связи между страницами,

- участники могут совершать действия над страницами,

- все действия участников и все операции над страницами сохраняются и используются для анализа динамики развития wiki-системы.

Наряду с приведенными особенностями построения модели wiki-системы, нужно отметить, что каждый участник этой системы обладает свойством ***age,*** характеризующим возраст участника. На начальном этапе формирования системы значение этой переменной равно нулю, а затем на каждом последующем цикле величина увеличивается на единицу.

Каждый участник обладает свойством ***retirement*** – значение, по достижению которого он перестает сотрудничать в системе. В случае, если значение ***age*** превосходит значение ***retirement***, тогда участник перестает действовать, но при этом участники не уничтожаются (становится невидимым).

Каждый участник обладает свойством ***status*** – статус, который может быть ***user*** или ***administrator.*** При этом участников в статусе администратор легко отличить на экране, они выделяются белым цветом, крупнее размером и действуют все время, пока функционирует система.

Каждый участник обладает свойством "***active?" true/false***– если участник достигает возраста старше retirement, то статус active? переключается на false.

С каждым участником связано несколько списков, которые представлены на рисунке 2:

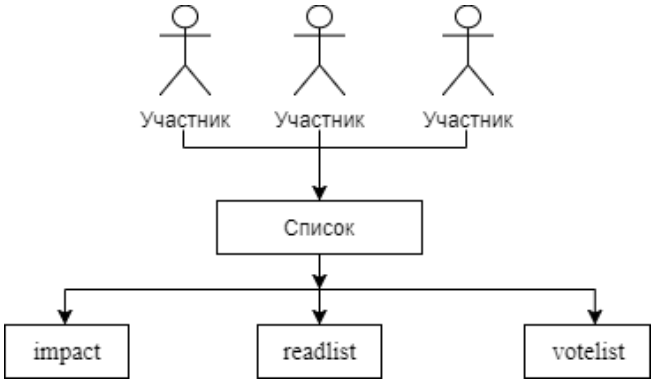


Рисунок 2 – Списки участников

где:

– *impact*– список, который позволяет внутри wiki-системы узнать все страницы, в создании и редактировании которых принимал участие этот автор;

– *readlist*– список, в который складываются прочитанные страницы;

– *votelis*t– список, в который складываются страницы, которые участник оценил«За»/«Против». По каждой странице участник голосует только один раз. По страницам, которые создал сам участник, он не имеет возможности голосовать.

Широкое повсеместное внедрение технических устройств привело к увеличению объема обрабатываемой информации, что позволило использовать агентно-ориентированные модели (АОМ) для решения различных коммерческих и технологических проблем. Это свидетельствует о необходимости применения специальных методов, например, таких как: оптимизация сети поставщиков и логистика; моделирование потребительского поведения (в том числе социальные сети); распределенные вычисления; ме-

неджмент трудовых ресурсов; управление транспортом; управление инвестиционными портфелями [2-5].

В этих и других приложениях стратегии поведения определяются с учетом поведения множества индивидуальных агентов-атомов и их взаимодействий. Таким образом, АОМ могут помочь в изучении влияния индивидуального поведения агентов на эволюцию всей системы в целом.

Одной из программ для разработки АОМ является бесплатно распространяемое приложение NetLogo. Изначально NetLogo был разработан как учебный инструмент, однако сейчас им пользуются не только студенты, но и тысячи исследователей. Это программа часто применяется в ВУЗах для обучения студентов основам АОМ. Схожей функциональностью обладает программа StarLogo.

Вывод. Инструментом для реализации более широкого спектра задачи в области АОМ является программа Swarm[en]. В ней используется язык программирования Objective-C и она может быть рекомендована программистам, использующих язык С, причем не только профессионалам, но и новичкам. Программировать в среде Swarm можно так же и на языке Java. Отметим также ещё несколько программ: MASON, Repast (используется Java), EcoLab (используется C++), Cormas (используется SmallTalk).

Список использованных источников

1. Хетагуров, Я.А. Проектирование автоматизированных систем обработки информации и управления / Я.А. Хетагуров. −М.: БИНОМ. Лаборатория знаний. 2015. −243 с.

2. Томакова, Р.А. Методологические основы научных исследований: учебное пособие / Р.А. Томакова, В.И. Томаков; Юго-Зап. гос. ун-т. − Курск, 2017. − 194 с.

3. Томакова, Р.А. Математическое обеспечение распознавания и классификации сложноструктурируемых биологических объектов/ Р.А. Томакова, Насер А.А., О.В. Шаталова// Международный журнал прикладных и фундаментальных исследований. 2012. −№4. –С.48-49.

4. Белобров, А.П. Нейросетевые модели морфологических операторов для сегментации изображений медицинских сигналов/ А.П. Белобров, С.А. Борисовский, Р.А. Томакова//Известия ЮФУ. Технические науки. 2010.№8(109). –С.28-32.

5. Борисовский, С.А. Нейросетевые модели с иерархическим пространством информативных признаков для сегментации плохоструктурированных изображений/ С.А. Борисовский, А.Н. Брежнева, Р.А. Томакова// Биомедицинская радиоэлектроника. 2010. №2. –С.49-53.

6. Томакова, Р.А. Методологические основы моделирования: учебное пособие/ Р.А. Томакова; Юго-Зап.гос.ун-т. – Курск, 2018. –258с.

7. Malyshev, A.V. Search of a Subcriber in a Reproduced-Behavior Program Multiconroller /A.V. Malyshev, M.V. Medvedeva, V.A. Koloskov //Telecommunications and Radio Engineering. 2004. Т. 62. № 4. С. 343-354.

8. Томакова, Р.А. Гибридные методы и алгоритмы для интеллектуальных систем классификации соложноструктурируемых изображений: автореф.дис. … докт.техн.наук: 05.13.17/Томакова Римма Александровна. – Белгород, 2013. –42с.

9. Rudomin, B. Hernandez, E. Millan, Fragment shaders for agent animation using finite state machines, In Simulation Modelling Practice and Theory Journal, Volume 13, Issue 8, Programmable Graphics Hardware November 2005, Pages 741—751 Elsevier/

10. Харзеева С.Э., Лушникова Е.И. Контекстуальное смысловое моделирование научного дискурса//Русский язык: исторические судьбы и современность . IV Международный конгресс исследователей русского языка: труды и материалы.-М.- 2010. -С. 144-145.

Севрюкова В.В., студент, e-mail: se_vv@bk.ru,
Алябьев С.А., студент, e-mail: dold4712@gmail.com,
ЮЗГУ, г.Курск, Российская Федерация

АНАЛИЗ ИСХОДНОГО КОДА ПРОЕКТА НА БАЗЕ ПРОЦЕССОРНЫХ МОДУЛЕЙ «БАГЕТ-83» И «БАГЕТ-83В» ДЛЯ РАЗДЕЛЕНИЯ ЕГО НА ДВЕ НЕЗАВИСИМЫЕ ЧАСТИ

В статье рассмотрены основные способы анализа исходного кода проекта для разделения проекта на независимые части. Приведены этапы анализа, а также рассмотрены существующие системы.

***Ключевые слова**: семантический анализ кода, лексический анализ кода, разделение проекта на независимые части.*

ANALYSIS OF THE SOURCE CODE OF THE PROJECT ON THE BASIS OF THE PROCESSOR MODULES "BAGET-83" AND "BAGET-83V" FOR SEPARATION IT FOR TWO INDEPENDENT PARTS

The article considers the main ways of analyzing the source code of a project for dividing the project into independent parts. In the article shows of the analysis source code, and also the existing systems are considered.

***Keywords**: semantic analysis of code, lexical analysis of code, division of the project into independent parts.*

Все существующие на данный момент проекты на базе процессорных модулей «Багет-83» и «Багет-83В» включают в себя две части: системное ПО и функциональное ПО. Однако одновременное содержание этих частей в одном проекте не всегда является удобным и безопасным, так как файлы функционального ПО используются в системном ПО, а файлы системного ПО используются в функциональном ПО. Из-за этого могут возникать ситуации, когда одна часть проекта изменяет файлы другой части. Кроме того, даже при незначительном изменении любой части (например, при добавлении нового функционала) приходится затрагивать также и вторую часть.

Поэтому необходимо из исходного проекта выделить две независимые части, и сформировать на основе них два независимых проекта: проект функционального ПО и проект системного ПО.

Для обеспечения доступа к необходимым файлам из другой части проекта необходимо произвести анализ кода исходного проекта и по результатам этого анализа сформировать дерево, отображающее зависимости между файлами, содержащими описание функций, и файлами, использующими эти функции.

Для анализа исходного кода проекта уже существует множество систем. Среди них выделяют:

1. Трансляторы – программы, переводящие входную программу на исходном языке на эквивалентную ей выходную программу на результирующем языке;

2. Компиляторы – трансляторы, переводящие исходную программу и эквивалентную ей объектную программу на язык машинных команд [1];

3. Интерпретаторы – программы, воспринимающие входную программу на исходном языке и выполняющие её, но, в отличие от трансляторов, не порождающие результирующую программу.

Важным этапом анализа исходного кода является лексический анализ [2]. На этом этапе происходит разбор водной цепочки на такие единицы, как ключевые слова, идентификаторы, константные значения и т.п. Такие единицы называются лексемами.

На этом этапе также происходит удаление комментариев и обработка директив условной компиляции.

Нередко для лексического анализа исходного кода используются различные регулярные выражения, которые позволяют не просто выделить необходимые лексемы, но и указать адреса их вхождений.

Результат работы лексического анализа является входными данными для синтаксического анализа. На этапе синтаксического анализа лексемы разбираются в соответствии с некоторой грамматикой, аналогичной грамматике используемого языка.

Не менее важным этапом анализа исходного кода является семантический анализ. Он необходим для проверки правильности текста исходной программы с точки зрения семантики входного языка, а также для преобразования текста, в соответствии с семантикой входного языка.

Для реализации семантического анализа исходного кода необходимо:

1. Проверить соблюдение во входной программе всех семантических правил входного языка. Как правило, это следующие проверки:

a. Каждая метка, на которую есть ссылка, должна присутствовать;

b. Каждый идентификатор должен быть описан только один раз;

c. Типы должны быть согласованы между собой;

d. Число фактических параметров должно быть согласовано с числом формальных параметров.

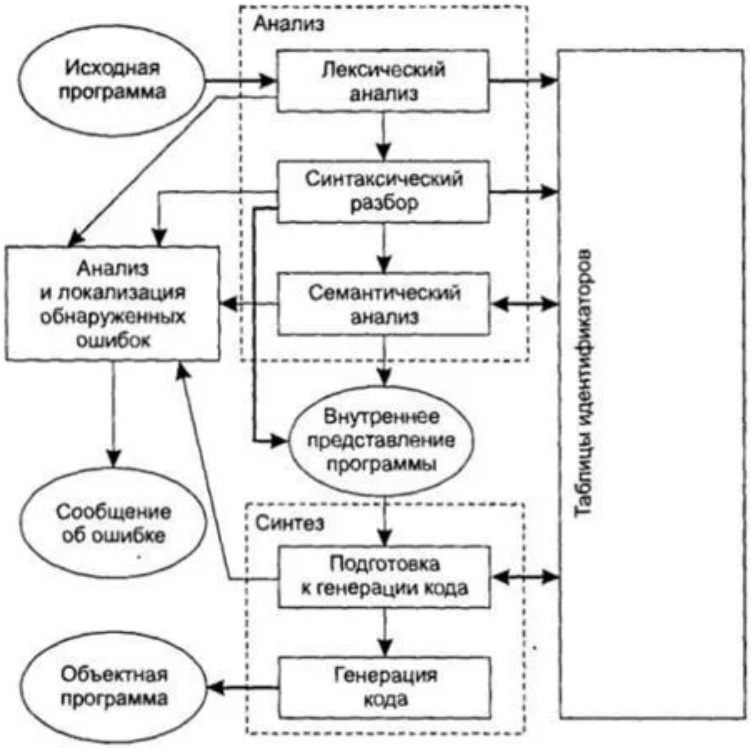


Рисунок 1 – Схема анализа исходного кода программы

В зависимости от используемого языка в этот список могут добавляться различные особенности этого языка.

2. Дополнить внутреннее представление программы в компиляторе операторами и действиями, неявно предусмотренными семантикой языка (в частности преобразование типов переменных в выражениях и при передаче их в качестве параметров в процедуры и функции).

3. Проверить элементарные смысловые нормы языков программирования, напрямую не связанных с входным языком программирования. Как правило, на этом этапе проверяются следующие смысловые нормы [3]:

a. Каждая объявленная переменная должна использоваться в программе хотя бы один раз;

b. Каждый цикл должен быть завершён;

c. Объявить переменную можно только один раз и т.д.

На рисунке 1 представлена схема анализа исходного кода программы.

Наиболее трудоёмкий процесс – составление грамматики исходного языка программирования, т.к. не для всех существующих языков, несмотря на продолжительность их существования, разработаны грамматики, точно отображающие все особенности этих языков.

Список использованных источников

1. Компиляторы: принципы, технологии и инструментарий [Текст] / А.В. Ахо [и др.]. – М.: ООО "И.Д. Вильямс", 2008. – 1184 с.

2. Вирт, Н. Разработка операционной системы и компилятора / Н. Вирт, Ю. Гуткнехт. – М.: ДМК Пресс, 2012. – 560 с.

3. Sebesta, R.W. Concepts of Programming Languages / R.W. Sebesta. – England: Pearson, 2016. – 785 с.

Научное издание

Сборник материалов
II Всероссийской научно-практической конференции
«**ПРОГРАММНАЯ ИНЖЕНЕРИЯ: СОВРЕМЕННЫЕ ТЕНДЕНЦИИ РАЗВИТИЯ И ПРИМЕНЕНИЯ**»
11-12 марта 2018 года

Ответственный редактор - *Томакова Р.А.*

Подписано в печать 16.05.2017 г.
Формат 60x84 1/16, Бумага офсетная
Уч.-изд. л. 7,6 Усл. печ. л. 6,8 Тираж 200 экз. Заказ № 558

Отпечатано в типографии
Закрытое акционерное общество "Университетская книга"
305018, г. Курск, ул. Монтажников, д.12
ИНН 4632047762 ОГРН 1044637037829 дата регистрации 23.11.2004 г.
Телефон +7-910-730-82-83