АНО «Институт логики, когнитологии и развития личности»
ООО «Базальт СПО»
НОУ «ИПС-Университет г. Переславля им. А. К. Айламазяна»
Институт Программных Систем РАН

Двенадцатая конференция «Свободное программное обеспечение в высшей школе»

Переславль, 27–29 января 2017 года

Сборник материалов конференции

Москва,
Basealt,
2017

Двенадцатая конференция «Свободное программное обеспечение в высшей школе»: Материалы конференции / Переславль, 27–29 января 2017 года. М.: Basealt, 2017. — 113 с. : ил.

В книге собраны материалы конференции, одобренные Программным комитетом двенадцатой конференции «Свободное программное обеспечение в высшей школе».

ISBN 978-5-9908979-1-5

Программа конференции

27 января, пятница

11.30-12.30 Регистрация участников и заселение

Дневное заседание
12.30–14.10

12.30 А. Е. Новодворский. Открытие. Информация оргкомитета

12.40 С. М. Абрамов. Приветственное слово

13.10–13.40 Н. Н. Непейвода

Курс «Представления чисел» 8

13.40–14.10 Д. К. Державин

Технологическое и методическое обеспечение учебных курсов Альт 9

14.10–15.40 Перерыв на обед

Вечернее заседание
15.40–18.00

15.40–16.10 С. В. Рощин

Развитие центра свободного программного обеспечения во Владимирском государственном университете: от преподавания и собственной инфраструктуры до регионального проекта GosLinux 11

16.10–16.40 М.Е.Рудаченко

Свободный эмулятор MARS в курсе «Архитектура ЭВМ и язык ассемблера» 16

28 января, суббота

Утреннее заседание
10.00–13.20

13.20–14.50 Перерыв на обед

Дневное заседание
14.50–17.40

16.20–16.40 Кофе-пауза

29 января, воскресенье

Утреннее заседание
10.00–13.10

11.30–11.50 Кофе-пауза

Дневное заседание
14.50–16.50

Вне программы

Николай Николаевич Непейвода, проф.
Переславль-Залесский, ИСП РАН, УГП им. А. К. Айламазяна

Курс «Представления чисел»

Аннотация

Для учащихся числа обычно даются лишь в одном стандартном представлении, что резко сужает кругозор, и не даёт возможность увидеть простые решения многих задач. В данном курсе систематизируются данные о представлении чисел, что дает возможность поставить массу задач для начинающих информатиков.

1. Принципиальная разница целых, рациональных и действительных чисел. Рациональные числа как частный случай алгебраического пополнения. Действительные числа как частный случай топологического пополнения.
2. Вред привычки рассматривать числа через конкретное представление. Деление в древнем Египте и в средние века: «инновационный регресс». Аликвотные дроби: проклятие школьников и математиков.
3. Представления целых чисел. Позиционные системы счисления с избыточностью и переменными основаниями. Их преимущества и недостатки. Система остаточных классов. Ситуация «нос вытащил — хвост увяз»: исключительно удобно производимые алгебраические операции и проблема со сравнением. Гибридные системы представления.
4. Представления рациональных чисел. Непрерывные дроби. Их преимуществка и недостатки. Округление рациональных чисел.
5. Представления действительных чисел. Безнадёжная невычислимость традиционных позиционных систем. Недостатки «машинных чисел». Непрерывные дроби для действительных чисел. Их различные формы. Их использование для точных вычислений.
6. А если учесть неточность? Интервальная арифметика. Системы счисления с перекрытием. Алгоритм Шворина.
7. Экзотика. Системы счисления с иррациональным основанием. Их преимущества.

Курс доступен также для школьников старших классов. Предполагается его пилотное прочтение в следующем учебном году в УГП.

Дмитрий Державин
Санкт-Петербург, ООО «Базальт СПО»
https://www.basealt.ru/

Технологическое и методическое обеспечение учебных курсов Альт

Аннотация

Задача прочтения учебного курса в области информационных технологий требует в настоящее время сочетания детальной проработки, применения современных технических и методических средств, а также возможностей быстрой адаптации к меняющимся требованиям отрасли.

Для решения этой задачи на курсах Альт успешно применяется ряд авторских методик, направленных на упрощение модификации учебной программы, автоматизацию поддержания её в целостном состоянии, а также на обеспечение слушателей возможностью максимально концентрироваться на восприятии материала, а не на его конспектировании.

Поддержка учебных курсов в области информационных технологий предполагает непрерывное решение следующих задач. Необходимо как можно более точно соответствовать быстро меняющимся требованиям отрасли. Это значит, что учебную программу придётся постоянно адаптировать. При этом не должна страдать целостность и последовательность изложения материала. Получение целостного набора знаний и навыков требует продолжительного времени для занятий теорией и практикой. При этом выделить на один учебный курс более 40 академических часов практически невозможно. Справиться с такими противоречивыми вводными помогает специальный комплекс мер.

Для построения подробного плана учебной программы используется представление её в виде ориентированного графа, для описания которого применяется язык «dot». Представление в виде графа помогает увидеть взаимосвязи между отдельными темами и привести программу к целостному и непротиворечивому состоянию. Выбор языка

dot обусловлен его лёгкой читаемостью и широкими возможностями обработки, принципиально позволяющими не только строить диаграммы, но и формировать произвольные текстовые представления. Метод помогает вносить изменения в программу и оперативно отслеживать при этом её целостность и внутреннюю непротиворечивость.

Для максимального приближения содержания учебной программы к требованиям отрасли применяется авторская разработка, базирующаяся на принципе «от обратного». Суть её в том, что сначала составляется список контрольных вопросов для проверки остаточных знаний и сформированных навыков, а затем, на основе содержания вопросов составляется, или, можно сказать, восстанавливается учебная программа. В комбинации с графовым представлением такой метод даёт возможность избежать включения в программу тем, не относящихся напрямую к задачам курса, а также помогает не забыть про темы, действительно необходимые для получения желаемого набора знаний и навыков.

Для объяснения теоретической части учебного материала применяется разработанная и успешно внедрённая в Санкт-Петербургском политехническом университете методика оформления рабочих тетрадей слушателей в виде «книжек-раскрасок» — кроме пространства для самостоятельного конспектирования они содержат схемы и диаграммы без подписей. Задача слушателей — в процессе занятия заполнить пробелы в соответствии с объяснением преподавателя. Такой подход позволяет слушателю сосредоточиться на содержании, а не на форме, и быстрее усваивать сложный материал за счёт «автоматизированного» конспектирования.

Практическая часть учебного материала рассматривается исключительно на примерах и в процессе объяснения демонстрируется на экране в аудитории. Видео с демонстрацией каждого примера вместе с голосом преподавателя записывается в файл, который слушатели по окончании занятия могут забрать с собой. Такой подход позволяет слушателям сосредоточиться на примере и не тратить много времени на запись увиденного в тетрадь, так как в случае необходимости скринкаст в любой момент можно будет воспроизвести. К тому же, в рабочей тетради, как было сказано выше, уже есть заготовки для ускоренного конспектирования. Методика также была разработана и опробирована в Санкт-Петербургском политехническом университете и продолжает экономить время и силы слушателей.

Здесь важно понимать, что одна из основных задач интенсивных курсов — дать слушателям как можно больше качественного и хорошо структурированного материала, предоставив им возможность потом к этим материалам возвращаться. При этом желательно избавить слушателей от выбора — успеть понять или успеть записать. С решением этих задач комбинация скринкастов и «конспектов-раскрасок» справляется хорошо.

При всей привлекательности скринкастов, основные практические навыки слушатели приобретают всё же на лабораторных работах. Поэтому важно дать им возможность в прямом смысле унести с собой как можно больше: конфигурационные файлы, историю команд, результаты выполнения заданий. Наилучший вариант здесь — унести с собой экземпляр установленной операционной системы, в которой выполнялись лабораторные работы. Эта задача решается с помощью загрузки ОС с live-flash с поддержкой сеансов. Образ для загрузочного носителя изготавливается в конфигурации, максимально адаптированной для конкретной слушательской аудитории. Это может быть образ дистрибутива сертифицированной ОС или специально составленный профиль для прочтения совершенно конкретного учебного курса.

Таким образом, мы приближаемся к решению основной задачи — сделать курс гибким, информативным, целостным и актуальным.

При этом, в соответствии с политикой открытости Альт, все материалы курсов, а также все инструменты, применяемые при подготовке и проведения курсов, распространяются на условиях свободных лицензий.

Сергей Рощин
Владимир, ВлГУ

Развитие центра свободного программного обеспечения во Владимирском государственном университете: от преподавания и собственной инфраструктуры до регионального проекта GosLinux

Аннотация

В целях решения практических, методических и организационных вопросов, связанных с реализацией государственной политики по импортозамещению, обеспечением экономической самодостаточности, государственного суверенитета и национальной безопасности в сфере информационно-коммуникационных технологий (ИКТ) во Владимирском государственном университете создан Центр компетенции по импортозамещению в ИКТ. В рамках центра действуют лаборатории свободного программного обеспечения (СПО) и GosLinux. Многолетние усилия по переходу с проприетарного на свободное ПО позволили добиться успехов во внедрении новых учебных курсов, частичного перевода собственной инфраструктуры ВУЗа на СПО, сформировать группы интересов и коллективы с необходимыми компетенциями. Важнейшим этапом развития СПО в регионе является успешный опыт внедрения СПО GosLinux в региональном отделении Федеральной службы судебных приставов (ФССП) во Владимирской области. На текущем этапе при поддержке администрации Владимирской области при участии ВлГУ рассматривается возможность масштабирования проекта на другие органы и структуры государственного управления с одновременной переподготовкой кадров.

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых» (ВлГУ) является ведущим региональным ВУЗом во Владимирской области. ВлГУ расположен в городе Владимире и имеет два филиала в городах Муром и Гусь-Хрустальный. Контингент учащихся ВлГУ по различным формам и направлениям исчисляется десятками тысяч человек. Кампус ВлГУ только лишь во Владимире, не считая различных вспомогательных объектов, включает 11 учебных корпусов и 13 общежитий.

В настоящее время задача обеспечения конкурентоспособности на рынке образовательных услуг не может быть решена без использования современных информационно-коммуникационных технологий (ИКТ) в управлении и научно-образовательной деятельности учебного заведения.

На текущий момент ВлГУ располагает современной информационно-коммуникационной инфраструктурой, которая содержит следующие ключевые компоненты:

- узел доступа к федеральной образовательной сети RUNNet и в Интернет, по двум гигабитным каналам связи с резервированием и балансировкой нагрузки (AS51352, BGP, IPv4/IPv6, университет располагает лицензиями оператора связи);

- высокоскоростная распределенная ЛВС с проводным и беспроводным доступом, обеспечивающая работу свыше 3,5 тыс. университетских устройств, а также до 10 тыс. пользовательских компьютеров одновременно;
- два центра обработки данных (ЦОД) с оборудованием HP ProLiant, IBM BladeCenter, Cisco UCS, NetApp (свыше 40 физических серверов) с реализацией концепций динамического ЦОД, виртуализацией (Hyper-V, VMware), средствами контроля и управления;
- научно-образовательных центр высокопроизводительных вычислений (НОЦ ВВ) — суперкомпьютер «СКИФ Мономах» с 64 вычислительными узлами с гибридной Linux/Windows архитектурой;
- значительный набор информационных сервисов, включая инфраструктурные сервисы DNS/Active Directory, System Center, Exchange, Skype for Busines и др., а также информационные сервисы для управления ВУЗом и обучения «Парус», «Галактика», IBM Lotus Domino, Moodle, Dspace и др.

Набор информационных сервисов, необходимых университету для управленческой и образовательной деятельности, и требования к ним постоянно растет. Поддержание информационной экосистемы университета только за счет использования проприетарного ПО является достаточно затратным в экономическом отношении при том, что на текущий момент есть возможность решения части задач с использованием свободно распространяемого программного обеспечения (СПО).

С перспективы ЦОД использование СПО и ПО с открытым кодом в ВлГУ осуществляется в следующих информационных системах:

- CentOS/RHEL версий 6 и 7:
  - организация контроля и учета доступа абонентов в Интернет (биллинг) на оборудовании HP ProLiant DL380 G5/Gen9,
  - личный кабинет абитуриента для подачи заявления в университет (виртуальный сервер);
  - управление АТС университета на базе решения «Протей»;
- FreeBSD версий 9 и 10 — серверы доступа в Интернет для абонентов в общежитиях (сервис mpd для терминирования PPPoE,

PPTP сессий), система блокировки запрещенных сайтов согласно перечня Роскомнадзор (SNORT, Quagga, Python и др.);

- SUSE Enterprise Linux/CentOS — суперкомпьютер «СКИФ Мономах», включая свободно распространяемые средства управления кластером XCat, torque, ganglia, библиотеки вычислений OpenMPI, MPICH, MVAPICH, MPICH2, средства разработки и тестирования gcc, BLAS, SCALAPACK.
- Linux Debian для семейства сайтов университета (`http://*.vlsu.ru`) — наряду с ОС и виртуализацией KVM используются CMS Typo3, MySQL, PosgreSQL и ряд других свободно распространяемых компонентов;
- Moodle (`http://dec.cdo.vlsu.ru/`) — используется для организации дистанционного обучения и взаимодействия с учащимися.
- DSpace (`http://dspace.org`) – организация электронной библиотеки внутривузовских изданий ВлГУ (`http://e.lib.vlsu.ru/`) и др.

Стоит отметить, что существенной проблемой использования бесплатных ОС на оборудовании корпоративного класса (серверы, СХД) является совместимость с оборудованием и техническая поддержка. Чаще всего производители оборудования обеспечивают поддержку только ОС RHEL/SUSE. В этой ситуации исключительно эффективным является использование технологий виртуализации: в ВлГУ реализовано частное облако на основе гипервизоров Hyper-V и VMware.

Что касается использования СПО для клиентской части ИТ инфраструктуры ВлГУ, прежде всего стоит упомянуть применение в качестве клиентских ОС различных сборок Linux. На текущий момент в образовательном процессе используются, напрмер, ОС CentOS, Debian, средства разработки Eclipse, OpenJDK, Oracle JDK и ряд других.

В рамках сотрудничества между Администрацией Владимирской области и ВлГУ при участии разработчика операционной системы Гослинукс компании «Ред Софт» на базе ВлГУ содан Центр компетенции Гослинукс. Для обеспечения работы Центра созданы лаборатория свободно распространяемого ПО и лаборатория Гослинукс.

Целью сотрудничества является продвижение и внедрение отечественного программного обеспечения среди пользователей органов государственной власти в качестве альтернативы зарубежным разра-

боткам, подготовка кадров для региона и повышение квалификации сотрудников.

Гослинукс является продуктом компании «Ред Софт» и был создан для Федеральной службы судебных приставов (ФССП). На текущий момент отделение ФССП во Владимирской области переведено на ОС Гослинукс.

В задачи Центра компетенции Гослинукс входит:

- подготовка учебных программ и программ повышения квалификации, организация и проведение образовательных мероприятий, в том числе, с использованием современных дистанционных технологий;
- изучение возможностей замены зарубежного проприетарного коммерческого ПО бесплатными аналогичными продуктами отечественного производства, что влечёт за собой снижение затрат органов государственной власти на программное обеспечение и повышает уровень доверия к используемым средствам обработки информации;
- анализ возможностей адаптации уже внедренных и используемых программных продуктов к работе в среде Гослинукс;
- разработка новых программных продуктов, способных функционировать в среде ОС Гослинукс.

Многолетние усилия по переходу с проприетарного на свободное ПО позволили ВлГУ добиться успехов во внедрении новых учебных курсов, частичного перевода собственной инфраструктуры ВУЗа на СПО, сформировать группы интересов и коллективы с необходимыми компетенциями.

Рудаченко Михаил Евгеньевич
Москва, ГБС РАН

Проект: Архитектура ЭВМ и язык ассемблера (Севастополь)
https://moodle.cs.msu.ru/enrol/index.php?id=42

Свободный эмулятор MARS в курсе «Архитектура ЭВМ и язык ассемблера».

Аннотация

Одной из заметных тенденций последних лет в построении курсов «Архитектура ЭВМ и язык ассемблера» (АЭиЯА)[7] стала практика использования эмуляторов для демонстраций и решения учебных задач.

В данной работе представлен подход к формированию основных требований к эмулятору. Дан краткий обзор существующих решений. Рассмотрен свободный эмулятор MARS (MIPS Assembler and Runtime Simulator)[3]. Обсуждаются вопросы возникшие в практике применения свободного эмулятора MARS в курсе «АЭиЯА».

При подготовке курса «Архитектура ЭВМ и язык ассемблера» (АЭиЯА)встал вопрос выбора эмулятора, поддерживающего архитектуру системы команд(ISA) MIPS[4]. Программных продуктов, реализующих ISA MIPS, существует большое количество. Эти системы довольно разнообразны по объёму и точности реализации ISA, объёму модулируемого оборудования, качеству работы. При выборе эмулятора необходимо учитывать и особенности учебного процесса. Для наших целей было важно учесть следующие из них:

- Работоспособность на разных платформах
- Покрытие материала курса
- Удобство для студента
- Автоматическая проверка заданий
- Низкий порог вхождения
- Приемлемость для преподавательского коллектива
- Лицензия

Доступные к настоящему моменту времени MIPS-эмуляторы можно разделить на несколько групп:

1. Образовательные SPIM[5], EduMIPS64[2], MARS.

2. Промышленные среды производителей микропроцессоров и QEMU.
3. Прочее.

В первой группе следует выделить SPIM и MARS, как программы специально ориентированные на курс Паттерсона, Хеннеси[1] [6]. EduMIPS64 примечателен удачной визуализацией работы конвейера, но реализует ISA MIPS64, что для наших целей избыточно. SPIM и MARS соответствуют всем выше перечисленным требованиям. Однако, MARS оказался более удобен для преподавательского коллектива.

Промышленные среды, как то MPLAB X, MCStudio и т.п., обладают рядом уникальных возможностей, но сложны в использовании, непрозрачны, часто непереносимы.

В третьей группе можно отнести множество представленных в сети интернет проектов разной степени зрелости. Объединяет их одно свойство — все они непригодны для использования в курсе АЭиЯА.

MARS написан на языке программирования java. Интерфейс программы типичен для java-swing и реализует следующие функции:

- интегрированный редактор и ассемблер.
- контроль скорости выполнения, в том числе пошаговое выполнение и возврат.
- обозреватель регистров общего назначения, регистров математического сопроцессора, специальных регистров и памяти.
- выбор формата отображения значений данных или адресов в десятичной или шестнадцатеричной системе,а для ячеек памяти и в ASCII.

Эмулятор MARS реализует методически важную часть инструкции MIPS. Возможно и использование псевдо-инструкций. Язык ассемблера компактен, но не более чем необходимо для закрытия всех тем учебного плана по ЯА.

В эмуляторе реализована простая модель памяти и модель исключений MIPS, что позволяет наполнить простыми примерами и задачами сложные темы курса о работе с внешними устройствами.

MARS может быть запущен в режиме командной строки, что позволяет реализовать автоматическую проверку учебных заданий. Аргументы командной строки используются для того,чтобы запросить вывод отдельных регистров, ячеек памяти или для проверки результатов работы программы.

Эмулятор оснащён набором инструментов для моделирования внешних устройств и управления ими. Утилиты: кеш, терминал, цифровая клавиатура, дисплей оказались очень полезны для иллюстрации соответствующих тем.

MARS спроектирован с учётом возможности самостоятельной модификации. Возможна реализация собственных команд ЯА и системных вызовов.

Достаточно просто могут быть реализованы собственные инструменты, так как инструмент представляет собой исполняемый в отдельном потоке класс, наследуемый от класса AbstractMarsToolAndApplication.

AbstractMarsToolAndApplication — абстрактный класс, который предоставляет общие компоненты для реализации инструмента.

Литература

[1] *Паттерсон Д., Хеннеси Д.*, Архитектура компьютеров и проектирование компьютерных систем, 4-е изд., СПб. Питер, 2012, 784 с., ISBN 978-5-459-00291-1

[2] EduMIPS64 Free cross-platform educational MIPS64 CPU Simulator, `http://www.edumips.org`

[3] MARS (MIPS Assembler and Runtime Simulator) An IDE for MIPS Assembly Language Programming, `http://courses.missouristate.edu/KenVollmar/Mars`

[4] MIPS32 Instruction Set Quick Reference, `https://imagination-technologies-cloudfront-assets.s3.amazonaws.com/documentation/MD00565-2B-MIPS32-QRC-01.01.pdf`

[5] SPIM: A MIPS32 Simulator, `http://spimsimulator.sourceforge.net`

[6] MARS: An Education-Oriented MIPS Assembly Language Simulator, `http://www.cs.missouristate.edu/~vollmar/MARS/fp288-vollmar.pdf`

[7] *Wolffe, G., Yurcik, W., Osborne, H. and Holliday, M.*, Teaching Computer Organization/Architecture With Limited Resources Using Simulators, ACM SIGCSE Bulletin 34

Михеев Андрей Геннадьевич
Москва, НИТУ МИСиС, ООО «Процессные технологии», RunaWFE
http://runawfe.org/

Курс обучения процессному управлению, использующий свободную систему RunaWFE

Аннотация

Разработан и апробирован в нескольких университетах курс обучения студентов процессной автоматизации предприятий. Практические задания курса выполняются в свободной системе управления бизнес-процессами RunaWFE. Использование свободного ПО для обучения позволяет легко внедрить курс обучения в учебный процесс любого российского ВУЗа. Также использование свободного ПО позволяет студентам выполнять домашние задания удаленно на домашнем компьютере, а преподавателям — удаленно готовиться к проведению занятий

Процессный подход к управлению

В соответствии с процессным подходом деятельность предприятия представляется в виде набора выполняющихся экземпляров бизнес-процессов. Бизнес процесс содержит набор узлов, соединенных между собой переходами. По этим переходам перемещаются точки управления. Узлы содержат задания, которые должны выполнить сотрудники или информационные системы.

Процессный подход позволяет повысить эффективность менеджмента путем формализации повторяющихся последовательностей действий, а также за счет возможности быстрого изменения бизнес-процессов в ответ на изменение условий бизнеса. Реализуют процессный подход системы управления бизнес-процессами и административными регламентами (СУБПиАР): В соответствии со схемой бизнес-процесса они раздают задания исполнителям и контролируют их выполнение.

В настоящее время СУБПиАР активно внедряются как в бизнесе, так и в государственных организациях, актуальной является задача обучения студентов экономических специальностей и специальностей, связанных с информационными технологиями, процессному подходу и работе с СУБПиАР.

Обучение студентов процессному подходу

В период с 2012 по 2016 г. на базе НИТУ МИСиС для обучения студентов процессной автоматизации был разработан курс процессного управления. Издано учебное пособие, содержащее теоретический материал [1], издан лабораторный практикум [2]. Составлен набор задач, используемых на зачетах и экзаменах (см. раздел «Задачи по процессному управлению» на ресурсе [3])

Курс был апробирован в НИТУ МИСиС, МЭСИ (в настоящее время — РЭУ им. Плеханова) и МГТУ им. Баумана. Занятия по отдельным разделам курса были проведены в УГАТУ, Финансовом университете, НИУ ВШЭ, Российском университете дружбы народов и МФТИ.

В рамках курса обучения студенты знакомятся с базовыми понятиями процессного подхода, в частности с понятиями «определение бизнес-процесса», «экземпляр бизнес-процесса», «исполнение экземпляра бизнес-процесса». В практической части курса отрабатываются вопросы построения схем бизнес-процессов, инициализации ролей бизнес-процессов, работе с внешними данными, построению форм заданий и взаимодействию с автоматическими исполнителями. Изучаются и отрабатываются на практике вопросы работы с переменными, сроками выполнения заданий, правилами выбора маршрутов движения точек управления. Также рассматриваются вопросы межпроцессного взаимодействия. Разработанные бизнес-процессы студенты исполняют под разными ролями в программной среде.

Использование свободного ПО с открытым кодом

В практических разделах курса используется свободное ПО с открытым кодом — система RunaWFE [4].

Использование в курсе свободного ПО позволяет студентам получать практические навыки работы с СУБПиАР, а также выполнять задания дистанционно, установив программное обеспечение на домашнем компьютере. Результаты выполнения заданий студенты выкладывают в кампус или посылают преподавателю по электронной почте. Преподаватель при этом тоже имеет возможность проверять решения на домашнем компьютере.

Курс обучения легко внедрить в любом российском ВУЗе, так как он использует только свободное ПО, доступное через интернет, не тре-

бующее оплаты или регистрации. В частности, это позволило опубликовать основные разделы курса на ресурсе Intuit.ru [5].

Свободное ПО также позволяет преподавателям различных ВУЗов свободно обмениваться разработанными бизнес-процессами без дополнительных затрат на приобретение ПО.

Система RunaWFE

Система RunaWFE является свободной СУБПиАР с открытым кодом. Ее дистрибутивы можно бесплатно скачать через интернет вместе с документацией и исходными кодами с портала sourceforge.net и использовать без каких-либо ограничений. RunaWFE распространяется на условиях открытой лицензии LGPL.

Инфраструктура проекта RunaWFE размещена в открытом доступе на портале разработчиков свободного программного обеспечения по адресу `http://sourceforge.net/projects/runawfe`. В 2016 году Система RunaWFE внесена в Единый реестр российских программ для электронных вычислительных машин и баз данных под номером 951 по классу ПО «системы управления процессами организации».

RunaWFE — это российский проект. Команда разработчиков находится в Москве, к разработчикам легко обратиться с вопросами, предложениями и пожеланиями.

Публикации [6-14] показывают, что система RunaWFE используется в ВУЗах России, Белоруссии и Украины как для обучения студентов, так и для организации внутренней деятельности ВУЗов.

Литература

[1] *Михеев А.Г.* Системы управления бизнес-процессами и административными регламентами на примере свободной программы RunaWFE. Второе издание. — М.: ALT Linux, 2016

[2] *Пятецкий В.Е, Михеев А.Г., Новичихин В.В.* Система управления бизнес-процессами: основы разработки бизнес-процессов с помощью свободного программного обеспечения: практикум — М.: Изд. Дом МИСиС, 2013.

[3] *Ссылка на сайт проекта RunaWFE:* `http://wf.runa.ru/`

[4] *Михеев А.Г., Орлов М.В.* Система управления бизнес-процессами и административными регламентами. // Программные продукты и системы, № 3, 2011

[5] *Михеев А.Г.* «Процессное управление на свободном программном обеспечении», — `http://www.intuit.ru/studies/courses/3529/771/info`

[6] *А.М. Кадан* Автоматизация документооборота кафедры в рамках концепции моделирования бизнес процессов, УО «Гродненский государственный университет имени Янки Купалы». Современные информационные технологии и ИТ-образование. Сборник избранных трудов VI Международной научно-практической конференции: учебно-методическое пособие. Под ред. проф. В.А. Сухомлина. — М.: ИНТУИТ.РУ, 2011.

[7] *Шеремет А. Н.* Моделирование и управление бизнес-процессами в Runa WFE : учебно-методическое пособие, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Кузбасская государственная педагогическая академия. — Новокузнецк : КузГПА, 2012.

[8] *Гареева А.П., Зверева Н.Н.* Система корпоративного планирования на платформе RunaWFE. Башкирский государственный аграрный университет. Уфа — V Международная студенческая электронная научная конференция «СТУДЕНЧЕСКИЙ НАУЧНЫЙ ФОРУМ 2013»

[9] *Е. А. Чичкарев, А.И. Симкин.* Мариуполь, Приазовский государственный технический университет Опыт внедрения свободного ПО в учебный процесс для специальностей факультета информационных технологий. Шестая конференция Свободное программное обеспечение в высшей школе Альт Линукс, 2011 с. 49–51

[10] *Кокунов В. А., Соколов Н. Е., Шарабаева Л. Ю.* Перспективы применения BPM-систем при развитии системы менеджмента качества современного вуза. Управленческое консультирование N 7 2014 с. 158 — 164 (Брянский филиал Финансового университета, Санкт-Петербургский филиал Финансового университета, Северо-Западный институт управления (Санкт-Петербург))

[11] *Точилкина Т.Е.* Моделирование исполняемых бизнес-процессов в СУБП RunaWFE в учебных проектах // Экономика и менеджмент инновационных технологий. 2015. № 9 (Финансовый университет. Москва)

[12] *Анна Вікторівна Марченко, Наталія Олександрівна Милостна* Формальний опис і автоматизація бізнес-процесу підприємства за допомогою ... Східно-Європейський журнал передових технологій, Vol. 4, Issue 2, 2011, pp. 28–31. (Сумський державний університет вул. Римського-Корсакова, 2, м. Суми, Україна)

[13] *Попова И.В.* Свободное программное обеспечение для управления бизнес-процессами. ФГБОУ ВПО «Магнитогорский государственный университет». Теория и практика применения свободного программного обеспечения: сборник трудов участников Всероссийской молодежной конференции с элементами научной школы/ Под общ. ред. Уметбаева З.М., Поповой И.В., Давлеткиреевой Л.З. — Магнитогорск : МаГУ, 2011.

[14] *Куликов Г. Г., Михеев А. Г., Орлов М. В., Габбасов Р. К., Антонов Д. В.* Создание BPMN-моделей в программном продукте RunaWFE. Лабораторный практикум по дисциплине «Автоматизированные информационные системы в экономике». — Уфа. УГАТУ. 2012

Александр Александрович Рыжов, Лев Николаевич Чернышов, Андрей Сергеевич Булыгин
Москва, НИУ МАИ, Финансовый университет

Система автоматической проверки ответов

Аннотация

Работа посвящена проектированию и реализации системы автоматической проверки тестовых ответов на естественном языке. Проверка ответов производится по смешанной схеме, включающей метод шаблонов и использование онтологических тезаурусов. Для построения шаблонов используется визуальный конструктор, упрощающий работу преподавателя. Для повышения качества поиска шаблонов используется механизм синонимов с использованием тезауруса RuTez. Описан процесс реализации каждого модуля системы и основные трудности. Разработка ведется на языке C++ с использованием библиотеки Qt. Весь исходный код является открытым программным обеспечением.

Автоматическое тестирование является неотъемлемой частью образования во многих ВУЗах. Но в большинстве случаев применяются закрытые тесты, в которых испытуемый узнает правильный ответ, а не проявляет способность логически мыслить. Наиболее привычным видом контроля является ответ на естественном языке. Тестируемый задает ответ в свободной форме и система автоматически сравнивает ответ с некоторым эталоном.

На данный момент реализовано множество систем с автоматическим тестированием, но немногие из них используют вопросы с открытыми ответами [1]. Авторы этих систем отмечают существенные

проблемы при их создании и невысокое качество. В данной работе ставится цель создать открытую и по возможности универсальную систему для автоматического тестирования с ответами на естественном языке, используя при этом максимально эффективные методы.

Проверка ответов базируется на нескольких основных методах: шаблоны, мешок слов, ролевые функции и некоторых других. В реализуемой системе используется «смешанная» схема с поддержкой шаблонов и онтологических тезаурусов. В случае шаблонов было решено не использовать напрямую регулярные выражения в интерфейсе для поиска совпадения, а реализовать специальный визуальный конструктор. Также реализуется возможность переключаться на прямое написание регулярных выражений с конструктора. Для повышения качества поиска шаблона используется механизм синонимов, где на основе первоначального строятся еще несколько автоматически сгенерированных шаблонов. К сожалению, шаблоны не смогут покрыть все варианты возможных ответов и для расширения возможностей в систему добавлен поиск семантической близости между понятиями эталонного ответа и ответа тестируемого. Для поиска семантической близости и механизма синонимов был использован тезаурус РуТез [3]. Определение семантической близости построено на том, что находится расстояние между вершинами в графе, построенном на основе тезауруса. Вершинами являются сами ключевые слова, а в качестве ребер связи «выше» — «ниже».

В системе предусматривается возможность создания различных видов тестов, в том числе закрытых.

Система может быть использована для любых предметных областей благодаря использованию шаблонов, но для качественной работы алгоритма семантической близости необходимо выбирать области, где даются четкие определения понятиям и имеются схожие формулировки в разных учебниках. В качестве примера можно привести курс «Право», который читается практически на всех факультетах технических ВУЗов. В данном курсе не используются математические формулы, что значительно облегчает обработку ответов.

Для подготовки базы тестов предлагается следующая технология. Каждый преподаватель, участвующий в создании тестов, готовит вопросы и, возможно, ответы к ним. При этом нежелательно, чтобы преподаватель видел ответы других преподавателей. По мере заполнения базы тестов, администратор начинает просматривать вопросы, на которые есть один или несколько ответов и начинает создавать по

ним шаблоны. Преподаватель может оценить вопросы определенным уровнем сложности и установить баллы за вопрос. Далее администратор оценивает ответы с помощью шаблонов, баллы по которым не должны существенно отличаться от выставленных преподавателем. После этого качество системы проверяется на студентах. При необходимости корректируются шаблоны и ответы. Все тесты хранятся в XML-формате и имеется возможность экспорта и импорта готовых наборов тестов. В настоящее время готовятся наборы тестов по дисциплинам «Операционные системы» и «Базы данных».

В настоящее время разработан прототип системы с разными типами вопросов и в том числе с открытыми ответами.

Литература

[1] *Мишунин О. Б., Савинов А. П., Фирстов Д. И.* Состояние и уровень разработок систем автоматической оценки свободных ответов на естественном языке. Современные наукоемкие технологии. — 2016 №1 — с. 38–44.

[2] *Фомин С.* WikiQuizzer или ВикиЭкзамены — тесты, удобные и для преподавателя, и для студента. Десятая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов / Переяславль, 24–25 января 2015 года. М.:Альт Линукс, 2015.

[3] *Добров Б. В., Лукашевич Н. В.* Тезаурус РуТез как ресурс для решения задач информационного поиска //Труды Всероссийской конференции ЗНАНИЯ-ОНТОЛОГИИ-ТЕОРИИ. — 2009. — Т. 1. — С. 250–259.

Сергей Мартишин, Владимир Симонов, Марина Храпченко
Москва, Институт системного программирования РАН, Российский государственный социальный университет, кафедра моделирования информационных систем и сетей факультета информационных технологий

Разработка информационных систем на основе серверной технологии Node.js

Аннотация

Представлены возможности серверной технологии Node.js в качестве программной платформы для написания JavaScript-приложений вне веб-браузера. Технология Node ориентирована на приложения с

высокой интенсивностью ввода-вывода и небольшое количество вычислений. Данная технология необходима для обучения студентов ИТ-направлений приемам работы с локальными и распределенными SQL и NoSQL-базами данных

В последние годы новые требования к подготовке студентов, обучающихся по направлениям «Информационные системы и технологии», «Информатика и вычислительная техника», «Прикладная математика и информатика» и ряду других, связаны с возрастанием объемов информации, в том числе и неструктурированной. Студентам необходимо изучить основы разработки распределенных информационных систем (ИС), основанных на различных системах управления базами данных (СУБД), как реляционных, так и NoSQL, куда входят серверные платформы, например, ИС предназначенные для реализации совместных проектов групп различных специалистов, в частности для диспетчерских пунктов отслеживания движения поездов, грузовиков, общественного транспорта в режиме реального времени и т.п.

Если наиболее популярными СУБД для создания распределенных ИС являются, среди СУБД реляционного типа, MySQL, MariaDB, PostgreSQL, а среди NoSQL-типа СУБД, такие как MongoBB, Redis, то среди платформ наиболее подходящей для обучения студентов является серверная технология Node.js, что объясняется простотой ей освоения и функциональными возможностями [1, 2]. К достоинствам платформы Node можно отнести то, что в ней язык JavaScript используется на стороне сервера, поэтому разработчики могут создавать веб-приложения для клиентской и серверной части на одном языке. Кроме того, широко распространенный формат обмена данными JSON является собственным форматом JavaScript и применяется в различных базах данных NoSQL (например, в MongoDB).

Node — платформа для написания JavaScript-приложений вне веб-браузера. Технология Node ориентирована на приложения с высокой интенсивностью ввода-вывода и небольшое количество вычислений.

Технология Node.js основана на JavaScript–движке V8 с открытым программным кодом, разработанным компанией Google, написанным на C++. Распространяется по лицензии BSD. Особенностями движка являются компиляция исходного кода JavaScript непосредственно в машинный код, эффективная система управления памятью за счет специальным образом организованной процедуры «сборки мусора» и позволяющая избежать утечки памяти, а также адаптивная оптимизация кода во время компиляции.

Базовый набор (ядро) модулей Node.js является компактным и предназначен для выполнения низкоуровневых задач. В него, в частности, включены модули для работы с файловыми системами (fs), сетями http, tls, https, `dgram` (для работы с UDPDatagram - сокетами, net (для создания TCP на локальном сервере), модуль dns для асинхронной обработки DNS-запросов, модуль `os` для вызова методов, связанных с операционной системой, модули для анализа ссылок и путей (`url`, `querystring`, `path`) и т.д.[2].

Большая часть функциональности Node обеспечивается модулями сторонних разработчиков, которые они в основном выкладывают на хостинге GitHub, Node-инструментария или реестре nmp (NodePackageManager) [3]. Условно модули могут быть разделены на классы, например, модули маршрутизации, для работы с реляционными/документными базами данных, модули шаблонов и пр. Для облегчения установки требуемых модулей в последние версии Node включают диспетчер пакетов nmp. При использовании nmp пользователю нет необходимости изучать установочные требования выбранного модуля (выбор конкретного модуля остается на усмотрение разработчика). Диспетчер пакетов произведет установку полностью, то есть установит также все модули, от которых зависит работа выбранного.

Заметим, поскольку любые операции ввода-вывода в Node.js по умолчанию реализованы как асинхронные и после их выполнения операции будет вызвана функция обратного вызова, то для корректной работы необходимо установить диспетчер пакетов nmp, например, модуль Async (предоставляет различные паттерны потока управления, например, async.parallel и async.waterfall).

Для удобства разработчиков Node.js предлагает различные платформы (например, Connect, Express, Geddy), предоставляющие широко используемые программные модули и набор функций для создания приложений. В настоящее время наиболее популярной платформой является фреймворк Express, поскольку он прост в использовании. Использование модуля `Express` позволяет быстро сгенерировать каркас приложения. Также ключевыми возможностями являются наличие гибкой системы маршрутизации запросов, перенаправления, а также собственная технология обработки ошибок [1]. Список необходимых модулей каждый пользователь может определить в файле package.json и установить их при помощи команды $ npm install.

Платформа Node.js в основном ориентирована на работу с NoSQL базами данных, например, Redis, MongoDB, CouchDB и многими дру-

гими. Одной из наиболее популярных документных СУБД NoSQL-типа является MongoDB, поскольку в MongoDB данные хранятся в виде документов формата BSON (двоичный JSON). Для работы с MongoDB (как и для других СУБД NoSQL-типа) существуют различные модули, которые необходимо установить дополнительно из соответствующего источника, например, указать название модуля mongodb в файле package.json. Он позволяет вставлять в текст скрипта стандартные команды для работы с MongoDB: создание, удаление, обновление, поиск и т.п. Обмен данными осуществляется через TCP. Однако Node.js поддерживает работу не только с СУБД NoSQL-типа, но и с реляционными базами данных. Для них также существует ряд модулей, которые устанавливаются дополнительно. Например, для СУБД MySQLимеется модуль mysql, который также прописывается в файле package.json.

Таким образом, изучение серверной технологии Node.js позволит студентам получить углубленные навыки создания распределенных ИС, что особенно актуально в связи с интенсивным процессом приведения в соответствие ФГОС ВО с новыми профессиональными стандартами. В результате выпускник должен обладать требуемыми профессиональными и профессионально-прикладными компетенциями, такими как способностью участвовать в создании компонент программного обеспечения, включая проектирование и администрирование локальных и распределенных баз данных, умению применять свои знания для проектирования и конструирования.

Литература

[1] *Пауэрс Ш.* Изучаем Node.js. Издательство: Питер, 2014 г., 398 с., ISBN: 978-1449323073 (англ.), 978-5-496-00356-8 (русск.).

[2] *Node.js.* [Электронный ресурс]. Режим доступа: URL: `https://nodejs.org/en` .Яз. англ. Дата обращения: 25.12.2016.

[3] *Nmp* [Электронный ресурс]. Режим доступа: URL: `https://www.npmjs.com`. Яз. англ. Дата обращения: 25.12.2016.

Денис Силаков
Москва, Virtuozzo / НИУ ВШЭ

Проект: Virtuozzo `https://cs.hse.ru/`, `https://virtuozzo.com/`

Преподаватель ВУЗа как посредник между студентами и разработчиками СПО

Аннотация

В докладе предлагается подход к организации студенческих проектов, при котором преподавателю отводится роль посредника между студентами и upstream-разработчиками различных СПО-проектов. Описывается опыт применения такого подхода при работе со студентами НИУ ВШЭ, выделяются его достоинства, недостатки и различные подводные камни, встреченные авторами за несколько лет применения подобной практики.

Вопрос привлечения новых разработчиков актуален для многих проектов СПО. При этом ВУЗы и прочие учебные заведения предоставляют неисчерпаемый источник потенциальных энтузиастов, которые могут пополнить ряды разработчиков. Основной же проблемой является неосведомленность многих студентов о том, насколько велик мир свободного ПО и какие возможности он предоставляет.

Помочь в решении этой нестыковки могут преподаватели и кураторы студенческих проектов. Вместо выполнения учебных задач, многие из которых остаются неизменными из года в год, имеет смысл привлекать студентов к решению реальных проблем, стоящих перед теми или иными свободными проектами.

Помимо погружения студентов в реальную жизнь, такой подход позволяет переложить часть нагрузки с преподавателя на апстрим. Во всех наших проектах подобного рода мы рекомендовали студентам общаться с апстримом напрямую (в публичных списках рассылки, системах учета ошибок и прочих публично доступных инструментах, используемых проектом) и только проводили регулярной мониторинг такого общения.

Для каких курсов это подходит?

Присоединение студентов к разработке незнакомого им продукта всегда требует определенного времени. Поэтому давать такие задачи

разумно только в рамках долгосрочных дисциплин — например, курсовых или дипломных работ либо предметов наподобие «Программного проекта» НИУ ВШЭ (который подразумевает работу команд студентов в течение нескольких месяцев над проектами конкретных заказчиков, в роли которых как раз и могут выступать различные СПО-сообщества).

Во многих ВУЗах студенты проходят двухнедельную производственную или технологическую практику. На наш взгляд, давать в рамках такой практики задачи, связанные с помощью некоторому свободному проекту можно только в том случае, когда студент сам неплохо знаком с этим проектом и готов ему помочь.

Как выбрать проект?

Чтобы выбрать проект, в который отправить студентов, преподавателю необходимо самому ориентироваться в мире СПО. Идеальными кандидатами являются проекты, в разработке которых преподаватель сам принимает участие, либо которыми он хотя бы пользуется. Поэтому первым делом познакомиться с проектом должен именно куратор со стороны ВУЗа — иначе возникнут трудности с точной формулировкой задачи, а впоследствии — с оценкой достигнутых результатов.

Например, в продуктах Virtuozzo используются открытые компоненты — в частности, ядро Linux, гипервизор KVM в связке с QEMU и множество других программ и библиотек для Linux. Неудивительно, что студенты, выполняющие задачи под руководством сотрудников Virtuozzo (например, в рамках прохождения курсов на соответствующей кафедре МФТИ), работают над этими продуктами и нередко их наработки отправляются в апстрим. В бытность автора сотрудником компании «РОСА», студенты выполняли немало задач для одноименных дистрибутивов Linux.

Если же говорить о выборе проекта, не имеющего непосредственного отношения к деятельности преподавателя, то следует обратить внимание на следующие аспекты:

- «адекватность» апстрима — существует немало проектов, внедриться в разработку которых человеку со стороны не очень просто, особенно если этот человек не имел подобного опыта;

- наличие у проекта внятного списка задач для реализации, среди которых есть такие, которые можно смело отдать студентам. Как правило, мы выбираем потенциально интересные для проекта задачи, к реализации которых долгое время никто не приступает.

Мотивация студентов

При выборе относительно больших задач (работа над которыми займет несколько месяцев) следует помнить, что студенты имеют право задуматься над вопросом – насколько выполнение задания пригодится им в будущем.

Не секрет, что многие учащиеся слабо знакомы с программированием под Linux и не очень-то жаждут исправить эту ситуацию (несмотря на то, что работа над открытым проектом даст им много навыков помимо собственно программирования — например, опыт работы в команде, использования популярных сервисов типа GitHub и так далее). Нынешние студенты гораздо охотнее занимаются веб-разработкой либо созданием приложений для мобильных устройств. Здесь их можно понять — рынок мобильных приложений существенно превосходит рынок настольных Linux-систем.

Однако следует помнить, что мир СПО вовсе не ограничивается традиционными Linux-приложениями. Например, в мире существует немало открытых программ для Andoid, а также мобильных версий популярных приложений для десктопа.

Примеры

Приведем примеры работ, предлагавшихся в 2015-2016 гг. студентам НИУ ВШЭ в рамках дисциплины «Командный проект»:

- Менеджер Bluetooth для LXQt;
- Android-клиент для системы сборки ABF;
- Универсальный инструмент подключения к облачным хранилищам (rosa-cloud-connector);
- Перевод QtRsynct на Qt5;
- Переиспользование учетных записей в Remmina;
- Веб-интерфейс для Koji на основе Bootstrap;

- Расширение форматов файлов, поддерживаемых QMMP;
- Инструмент локализации desktop-файлов с использованием online-сервисов автоматического перевода.

Среди указанных примеров (да и не только них), интерес вызвали только задачи, имеющие отношение к веб- или мобильной разработке. Из прочих проектов до стадии отправки в апстрим дошел только менеджер Bluetooth для LXQt. С учетом такой тенденции, в будущем мы планируем при выборе целевых проектов сместить акцент с приложений для рабочего стола на мобильные продукты и веб-сервисы. Надеемся, что в итоге подобные задания для студентов принесут пользу как самим обучающимся, так и сообществу СПО.

Иван Анатольевич Хахаев
Санкт-Петербург, Университет ИТМО

Особенности использования свободных программ на занятиях в магистратуре

Аннотация

Описывается применение свободных программ для решения задач на практических занятиях в магистратуре Университета. Приводится типовой перечень классов задач и соответствующих свободных программ. Оцениваются результаты использования указанных программ в учебном процессе.

В осеннем семестре 2016/2017 учебного года проводились практические занятия в магистратуре по направлениям 12.14.02 «Оптотехника» (15 человек) и 38.04.01 «Экономика» (23 человека) по дисциплинам «Компьютерные технологии в инновационной и педагогической деятельности» и «Компьютеры технологии в экономической науке и производстве» соответственно.

Применение свободных программ преследовало этические и методические цели. Этические аспекты подробно разобраны с статье Р. Столлмана (https://www.gnu.org/education/edu-schools.ru.html). В методическом аспекте важно было вывести студентов из «зоны комфорта», активизировать исследовательские способности и показать наличие более одного инструмента для решения каждого класса задач.

При проведении практических занятий использовалась операционная система ALT Linux KDesktop (p7 с обновление до p8).

Для студентов направления подготовки «Оптотехника» (дисциплина «Компьютерные технологии в инновационной и педагогической деятельности») в течение 68 часов аудиторных занятий рассматривались следующие темы:

- технологии создания интеллект-карт (FreeMind)
- технологии управления проектами (ProjectLibre, GanttProject)
- технологии создания презентаций в отсутствие офисного пакета (Reveal.js и LaTex/Beamer)
- технологии создания учебных видеороликов (Blender)
- технологии создания элементов курсов дистанционного обучения (Moodle).

Следует отметить, что решение с Reveal.js студенты нашли самостоятельно, подготовка презентации заняла около 6 часов, однако на создание презентации в LaTex/Beamer в среднем ушло 2 часа. Вопреки ожиданиям никто из 15 студентов не создал сколько-нибудь законченную динамическую 3D-модель.

Для студентов направления подготовки «Экономика» (дисциплина «Компьютерные технологии в экономической науке и производстве») в течение 51 часа аудиторных занятий планировалось рассмотреть следующие темы:

- реализация основных экономико-математических методов (LibreOffice Calc)
- технологии автоматизации финансовых вычислений (LibreOffice Calc)
- технологи создания функциональных моделей IDEF0 (LibreOffice Draw)
- технологии создания интеллект-карт (FreeMind)
- технологии управления проектами (ProjectLibre, GanttProject)
- технологии моделирования бизнес-процессов (RunaWFE).

К сожалению, последний пункт оказался не реализован в осеннем семестре вследствие неработоспособности текущей версии RunaWFE.

По итогам проведенных занятий можно данный опыт признать положительным и сделать следующие выводы:

- технологии создания интеллект-карт оказалась новой для студентов, но признана полезной и интересной
- все основные экономико-математические методы (оптимизация, регрессии, задача межотраслевого баланса) успешно реализуются средства LibreOffice Calc
- обнаружилась неработающая финансовая функция ODDFYIELD() (ДОХОДПЕРВНЕРЕГ())
- неожиданное отсутствие результатов с Blender связано с отсутствием мотивации к трудной и кропотливой деятельности
- «профессиональные» свободные программы в составе дистрибутивов (RunaWFE, R, DRAKON, QGIS) требуют тестирования преподавателями перед началом использования в учебном процессе.

Анвар Галимович Мифтиев, Иван Анатольевич Хахаев
Санкт-Петербург, Консорциум «Кодекс», Университет ИТМО

Методика обучения LibreOffice на опыте внедрения офисного пакета в Консорциуме «Кодекс»

Аннотация

Описываются проблемы, выявленные в ходе мониторинга реализации проекта миграции делопроизводства в ИТ-компании на свободный офисный пакет. Рассматриваются различные категории проблем и варианту путей их решения. Делается вывод о необходимости корректировки методики переподготовки сотрудников Консорциума.

В 4-м квартале 2015 года в консорциуме «Кодекс» было завершено обучение 30% сотрудников применению LibreOffice (LO) в порядке реализации проекта миграции на свободный офисный пакет. Обучение проводилось в режиме экспресс-курсов (6 часов), направленных в основном на освоение LO Writer. В 4-м квартале 2016 года была проведена инвентаризация применения офисных пакетов на рабочих местах сотрудников Консорциума.

В результате обследования 415 рабочих мест (PM) было выявлено, что 12% сотрудников в качестве основного офисного пакета используют LibreOffice, остальные используют в качестве основного офисного пакета Microsoft Office (MSO) различных версий (за исключением

отдельных сотрудников, вообще не использующих никакие офисные пакеты).

Среди причин применения MSO в качестве основного офисного пакета выявлены следующие.

1. Отсутствие информации о наличии LO на PM или о возможностях LO (в частности, сохранение и чтение форматов MSO 97-2003 и MSO 2007-2013) (категория «Нет знаний»).
2. Привычка (наработанные приёмы) использования MSO или автоматическое открытие файлов форматов MSO средствами MSO (категория «Привычка»).
3. Проблемы совместимости текстовых документов, электронных таблиц и презентаций при обработке документов, полученных от других пользователей или внешних контрагентов (партнёров), ошибки верстки в файлах PDF (категория «Совместимость»).
4. Производственная необходимость, включающая следующие факторы, явно указанные пользователями (категория «Производственная необходимость»)
   a) Разработка продуктов для клиентов, заведомо использующих исключительно MSO (в частности, госорганов)
   b) Тестирование взаимодействия продуктов с различными версиями MSO
   c) Обеспечение поддержки формата mht для всего жизненного цикла подготовки он-лайн справки продуктов Консорциума
   d) Обеспечение работы с файлами по протоколу WebDAV («облачное» хранилище)
   e) Обеспечение обработки текстовых документов и файлов электронных таблиц макросами
   f) Интеграция панелей инструментов продуктов Компании и продуктов конкурентов в окно офисного пакета
   g) Интеграция с внутрикорпоративными информационными системами.

Было выявлено несколько категорий причин продолжения использования MSO в качестве основного или дополнительного офисного пакета. Список условных названий таких причин следующий:

- «Нет знаний»

- «Привычка»
- «Совместимость»
- «Производственная необходимость».

Категория «Нет знаний» (2%) является достаточно малочисленной и пользователи из этой категории потенциально перераспределяются между остальными тремя категориями.

Пользователи категории «Привычка» (25%) могут быть переведены в группу активных пользователей LO простым организационно-техническим решением (рассматривается вариант установки бесплатных программ для просмотра и печати соответствующих файлов MSO).

Наибольшие трудности при дальнейшей миграции ожидаются для категории пользователей, названной «Производственная необходимость» (30%).

Решение части проблем совместимости (искажения внешнего вида документов при получении или отправке) для пользователей из категории «Совместимость» (43%) возможно по нескольким направлениям:

- актуализация установленных версий LO по крайней мере до релиза 5.2
- обучение пользователей контролю за шрифтами, используемыми в документах MSO и LO (минимальные проблемы с отображением обеспечиваются при использовании шрифтов Times New Roman, Arial и Courier New)
- обучение пользователей использованию PDF для документов, не подлежащих редактированию контрагентами (или PDF с полями для заполнения), также разумной представляется публикация презентаций в PDF штатными средствами LO
- разработка шаблонов типовых документов, обучение пользователей применению стилей при создании документов и для быстрой правки форматирования элементов документа (если в документе параметры установлены в явном виде в миллиметрах и наборы шрифтов синхронизованы, то возможности искажения существенно снижаются).

Снижение производственной необходимости применения MSO возможно как естественным путем при миграции госорганов на россий-

ские и открытые офисные пакеты, так и реализацией внутренних проектов по изменению инфраструктуры и технологий разработки продуктов Консорциума.

Также следует отметить устойчивое убеждение сотрудников, что «везде учат только MSO», сложившееся на основе личного опыта.

Выводы:

1. Методика обучения LibreOffice в Консорциуме должна быть скорректирована с учетом выявленных проблем, поскольку часть проблем в миграции связана исключительно с обучением пользователей

2. Экспресс-курсы длительностью в 6 часов, организованные в начале процесса, могут играть роль ознакомительных и не обеспечивают подготовку, требуемую для решения производственных задач. Однако такие курсы являются хорошим стартом для пользователей, имеющих мотивацию к самообучению

3. Проявилась потребность в централизованном управлении конфигурациями LibreOffice (клонирование настроек на большое количество рабочих мест)

4. Проявилась необходимость в проблемно-ориентированном обучении, которое может быть реализовано в виде консультаций конкретных пользователей по конкретным проблемным вопросам.

5. Требуется мониторинг и актуализация имеющихся в Консорциуме методических материалы по компонентам LibreOffice (Writer, Calc, Draw, Impress, Base), поскольку при изменении версий происходят заметные изменения в интерфейсе и функциях пакета.

Олег Ивченко, Алексей Драль
Долгопрудный, Москва, Московский физико-технический институт (государственный университет)

Проект: Образовательная IDE «КуМир»
`https://github.com/victor-yacovlev/kumir2`

Система HJudge или как автоматизировать проверку заданий при изучении работы с большими данными

Аннотация

Тестирование приложения для обработки больших массивов данных в большинстве случаев сводится к запуску приложения на небольших тестовых выборках. Однако если мы имеем дело с большими данными, то даже тестовая выборка может занимать несколько гигабайт. Соответственно, результат работы приложения тоже может быть большим. Проверка такого результата на корректность становится трудоёмким процессом. Тестирование Hadoop-приложений включает проверку (1) корректности сборки приложения, (2) успешности его запуска на Hadoop-кластере, (3) результата работы. В рамках курсов обработки больших данных обычно затрагивается несколько сервисов экосистемы Hadoop, что увеличивает сложность создания единого интерфейса для тестирования таких приложений. В применении к высшей школе, при тестировании приложений нужно учесть и то, что разные ошибки в коде по-разному влияют на результат. Иными словами, в целях более справедливого оценивания работ студентов нужно иметь богатый набор тестов для идентификации ошибок программы. Существуют различные программные продукты, решающие одну или несколько из описанных проблем. Однако не найдено системы, которая бы решала все эти проблемы одновременно. Это создало предпосылки для разработки своего продукта — системы `HJudge`, о которой и пойдёт речь в докладе.

Вступление

В рамках любого образовательного технического курса незаменимую роль играют практические задания. Они дают возможность студентам попрактиковаться в пройденном материале, а преподавателям — оценить их навыки. Однако проверка заданий является довольно

рутинной работой и при большом количестве участников курса требует серьёзных затрат времени. Отсюда вытекает необходимость автоматизации этого процесса.

Курс ХОБОД

Хранение и обработка больших объёмов данных (или BigData) — это курс, который читается для студентов 1-го года магистратуры ФИВТ МФТИ с 2015 г. Курс призван дать начальные знания в области хранения и обработки данных, для работы с которыми не достаточно одной машины [1]. Практическую часть курса составляют программы, разрабатываемые с использованием сервисов экосистемы Hadoop (Hadoop, Hive, Spark, HBase). Чтобы исключить возможность решения заданий обычными средствами, студентам выдаются большие датасеты (несколько десятков гигабайт).

Процесс проверки заданий

Студент оформляет пост в системе Piazza [2] по определённым правилам и выкладывает свою программу на Hadoop-кластер. Далее преподаватель собирает программу и запускает её. Во время работы программы с помощью интерфейса *JobTracker* [3] наблюдается состояние её выполнения. После отработки приложения на кластере проверяются его результаты. Последняя стадия проверки — это *Code review*, в рамках которого проверяется грамотность написанного кода. Такой метод проверки задач имеет свои преимущества при небольшом количестве заданий.

Автоматизация проверки заданий

Процесс проверки заданий, описанный выше можно свести к тестированию Hadoop-приложений. Тестирование включает проверку:

1. корректности сборки приложения,
2. успешности его работы на Hadoop-кластере,
3. результата работы.

Существует много библиотек, способных решить отдельно каждую из проблем. Например, этап (1) можно провести с помощью Maven [4].

Этап (2) можно провести с помощью инструментов, входящих в поставку Hadoop [5]. Проверить результат работы приложения теоретически можно и с помощью утилиты `diff`. Но из-за большого размера (а иногда и количества) выходных файлов это практически невозможно. Кроме того, в рамках курсов обработки больших данных обычно затрагивается несколько сервисов экосистемы Hadoop, что увеличивает сложность создания единого интерфейса для тестирования приложений.

Единой системы, которая бы покрыла все описанные выше проблемы, на данный момент не существует. Это послужило толчком к разработке `HJudge`.

Система HJudge и её применение при проверке заданий

Система `HJudge` — это программа, написанная на Python 2.7 и использующаяся для тестирования приложений, разработанных на Hadoop, Hive, Spark, HBase. Входными данными системы являются параметры запуска одного или нескольких тестируемых заданий. Они могут вводится в виде аргументов командной строки, либо с помощью конфигурационного файла. В качестве результата система выдаёт отчёт о проведенном тестировании.

В первом приближении архитектура `HJudge` состоит из основной части и расширяемого набора встраиваемых тестов. Основная часть состоит из ядра и нескольких модулей, каждый из которых отвечает за определённый этап работы приложения:

- модуль обработки входных данных,
- модуль валидации сборки,
- модуль проверки корректности запуска.

Тесты для данных модулей уже разработаны и поставляются вместе с системой. В большинстве случаев их достаточно, однако есть возможность встраивать свои.

Также существует отдельная компонента, отвечающая за проверку результата работы приложения. В эту компоненту можно встраивать тесты трёх типов:

- Построчные тесты — проверяют корректность отдельно взятой строки вне зависимости от других,
- Пофайловые тесты — проверяют корректность одного файла,

- Тесты на сравнение с эталоном — сравнивают фрагменты данных результата с фрагментами данных, сгенерированными эталонным приложением.

Тест представляет собой обычную Python-функцию. Для встраивания теста в систему достаточно добавить эту функцию в класс, соответствующий типу теста. По умолчанию система настроена так, что тестирование приложения завершается если не пройден хотя бы один тест. Это экономит время и ресурсы тестирования.

Ещё одной замечательной особенностью `HJudge`, облегчающей проверку заданий, является возможность тестирования сразу нескольких заданий. Она реализуется с помощью конфигурационного файла, в котором указываются данные о тестируемых заданиях.

На данные момент исходные коды системы `HJudge` ещё не выложены в открытый доступ, они хранятся на кафедральном репозитории.

Планы по дальнейшему развитию

1. Интеграция с платформой Piazza. Поскольку студенты работают с Piazza, планируется интегрировать `HJudge` с ней — реализовать автоматическую передачу информации о задании с Piazza в систему.
2. Интеграция с облачной платформой Everest [6]. Запуск системы как web-сервис позволит пользоваться ею не только преподавателям, но и студентам, поскольку можно будет ограничить доступ к тестам.
3. Реализация возможности добавлять тесты, написанные на языках, отличных от Python.

Выводы

1. Система `HJudge` успешно проявила себя на курах ХОБОД и «Многопроцессорные вычислительные системы» в 2016 г., в качестве проверяющей системы, позволив уменьшить время проверки заданий в среднем в 8-10 раз.
2. В 2016 г. `HJudge` была поставлена на баланс кафедры АТП ФИВТ МФТИ.

Литература

[1] *Ивченко О. Н., Драль А. А.* Hjudge: система тестирования приложений в экосистеме Hadoop // Сборник научных трудов МФТИ «Модели и методы обработки информации», г. Долгопрудный: МФТИ (ГУ), 2016, — 112–122.

[2] *Piazza.* The incredibly easy, completely free Q&A platform, URL: `http://piazza.com`. — Название с экрана.

[3] *Tom White*, Hadoop: The Definitive Guide, издание 3, O'Reilly, 657.

[4] *Maven.* Introduction to the Build Lifecycle, URL: `https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html`. — Название с экрана.

[5] *Hadoop Wiki.* How to develop Hadoop Tests, URL: `https://wiki.apache.org/hadoop/HowToDevelopUnitTests`. — Название с экрана.

[6] *O. V. Sukhoroslov, A. O. Rubtsov, S. Yu. Volkov.* Development of distributed computing applications and services with Everest cloud platform // Computer Research and Modeling, г. Москва: ИППИ РАН, 2015, — 593–599.

А. Г. Кушниренко, Я. Н. Зайдельман, В. В. Тарасова, А. Г. Леонов
Москва, ФГУ ФНЦ НИИСИ РАН

Информатика 7-9

В 2017 году в издательстве «Дрофа» выходит новый комплект учебников информатики для основной школы. Учебники разработаны авторским коллективом под руководством А.Г.Кушниренко.

Комплект соответствует государственным образовательным стандартам, рассчитан на трёхлетний курс обучения (с 7 по 9 класс) и состоит из трех учебников — по одному на каждый год обучения.

Новые учебники продолжают линию, начатую в 1985 году первым массовым отечественным учебником информатики под редакцией академика А.П.Ершова: информатика рассматривается как естественно-научная дисциплина; математика, программирование и технологии — как основания этой дисциплины; в процессе освоения

предмета ученики должны решить значительное количество не слишком сложных, но содержательных задач.

Основные особенности новых учебников.

1. Учебники XXI века

Когда-то школьникам приходилось объяснять, что такое компьютер и для чего он нужен. Сегодня практически у каждого ученика есть смартфон или планшет или ноутбук, а часто и несколько подобных устройств. Сегодняшнему школьнику не нужно объяснять, что такое социальная сеть или поисковая система, не нужно объяснять, для чего компьютер нужен лично ему.

С другой стороны, эта ситуация как никогда требует разумного использования техники и понимания основных принципов работы с ней. Это необходимо, чтобы сохранить необходимую конфиденциальность и уберечься от мошенников, чтобы эффективно использовать имеющиеся возможности и приобретать технику, адекватную потребностям.

В новых учебниках сделана попытка приблизиться к современному школьнику. Авторы не делают вид, что ученик ничего не знает об окружающем его компьютерном мире. Наоборот, они все время подчеркивают, что многие вещи ученикам уже знакомы, а учебник помогает понять, как это все устроено, как работает, как эффективнее этим пользоваться.

2. Неформальное изложение математических основ информатики

Элементы комбинаторики, теории множеств и математической логики изложен в учебниках неформально, но достаточно глубоко, по схеме, предложенной М.А. Ройтбергом: разбираем естественные вопросы, решаем ключевые задачи, не вводим без необходимости формальных определений и обозначений. Кроме того, по возможности увязываем теорию множеств и математическую логику с уже освоенной практикой работы с условиями и логическими операциями алгоритмического языка.

3. Содержательное изучение программирования

Авторы считают, что провозглашенный когда-то тезис о необходимости развивать алгоритмическое мышление нельзя считать устаревшим. Поэтому в учебниках последовательно проводится алгоритмический подход.

Действующая в настоящее время «Примерная программа по информатике для основной школы», одобренная решением федерального учебно-методического объединения по общему образованию (протокол от 8 апреля 2015 г. № 1/15) рекомендует для изучения в основной школе классический минимальный набор понятий последовательного императивного программирования: имена и значения переменных, типы данных, оператор присваивания, массивы, ветвления, циклы, процедуры. Однако примерная программа не фиксирует ни глубину изложения этого материала, ни объем, отводимый ему в учебнике. Авторы сочли необходимым изложить этот материал достаточно детально и в каждом из 3 учебников вопросы, так или иначе связанные с программированием, занимают до половины всего объема.

При этом сделана попытка, не углубляясь в профессиональное программирование, которое, разумеется, не должно быть частью общего образования, избежать и примитивного подхода, при котором все ограничивается изучением простейших синтаксических конструкций конкретного языка, без обсуждения содержательных примеров и проведения какого либо систематического тренинга.

Поэтому авторы учебника ограничились ровно тем набором понятий последовательного программирования, который предусмотрен примерной программой — никакие дополнительные вопросы, например, сложные структуры данных или подходы объектного программирования в учебниках не затрагиваются. Но, по мнению авторов, элементы программирования, предусмотренные программой, должны быть усвоены твердо. В частности, одним из важнейших разделов Примерной программы авторы считают «Примеры задач обработки данных». В этом разделе приведены примеры задач, которые должны уметь решать ученики по завершению курса:

- нахождение минимального и максимального числа из двух, трех, четырех данных чисел;
- нахождение всех корней заданного квадратного уравнения;

- заполнение числового массива в соответствии с формулой или путем ввода чисел;

- нахождение суммы элементов данной конечной числовой последовательности или массива;

- нахождение минимального (максимального) элемента массива.

Авторы учебника стремились сделать для учеников, освоивших курс, перечисленные и им подобные задачи рутинными. Учебники и авторская программа составлены в предположении, что в течение каждого года обучения каждый ученик, работая в классе и дома, выполнит несколько десятков простых упражнений и задач, требующих написания одного-двух десятков строк кода.

Но изучение программирования не сводится к рассмотрению базовых понятий и задач. Все эти понятия и программирование в целом рассматриваются как инструмент для решения содержательных задач. Ученики должны понимать, что главное — не применить цикл или заполнить массив, а решить задачу, и любые циклы и массивы нужны не сами по себе, а как средство для решения задачи.

В соответствии с этим подходом во все темы курса так или иначе вносится алгоритмическое содержание. Например, изучение файловой системы компьютера сопровождается описанием программных инструментов работы с файлами, рассмотрение систем счисления завершается построением формальных алгоритмов перевода. В разделе «Моделирование» в учебнике 9 класса, рассматриваются сравнительно сложные содержательные задачи, для решения которых не требуется специальных программистских знаний сверх школьного курса, но необходима логика, алгоритмическая культура, умение анализировать задачу и выстраивать последовательность действий. Это, например, такие задачи, как приближенное решение уравнений методом половинного деления, вычисления методом Монте-Карло, моделирование физических процессов (на примере задачи о вытекании воды из кубического бака). Решение каждой из этих задач в тексте учебника доводится до работающей программы, а упражнения посвящены модификациям этих программ для решения аналогичных задач.

4. Безальтернативная ориентация на школьный алгоритмический язык и учебную систему КуМир

Все программирование в учебниках ведется на школьном алгоритмическом языке, предполагается использование свободно распространяемой системы программирования КуМир. Учебники тесно интегрированы с алгоритмическим языком и с системой КуМир. В учебниках 7 и 8 класса интенсивно используются исполнители КуМира Робот и Чертежник, система КуМир регулярно упоминается в учебниках всех трех классов, многие иллюстрации сделаны на основе скриншотов системы, оформление программ в тексте сделано точно таким же, как в окне редактора КуМира.

В процессе работы над учебниками авторы тесно сотрудничали с разработчиками КуМира, по просьбе авторов в систему были внесены многочисленные правки и усовершенствования. Эта работа продолжается, предполагается, что новые версии КуМира в течение ближайших нескольких лет будут сохранять стопроцентную совместимость с учебниками.

КуМир был выбран авторами по нескольким причинам.

Мы считаем программирование в первую очередь инструментом для решения задач, а инструмент не должен отвлекать слишком много сил на свое освоение. КуМир очень прост в освоении и использовании, он позволяет новичку на первом же занятии полностью решить (то есть, набрать текст программы, запустить, отладить, получить результат) несколько простых, но вполне содержательных задач.

КуМир содержит развитые средства отладки, удобные и понятные для начинающего, простой русскоязычный интерфейс, возможность проектирования и выполнения заданий и курсов (практикумов) с автоматизированной оффлайн проверкой. Все это делает его удобной учебной системой, более эффективной при обучении, чем доступные сегодня профессиональные среды.

Встроенный в КуМир школьный алгоритмический язык — простой, но достаточно богатый язык программирования. Хотя по набору понятий (выразительной силе), алгоритмический язык похож на Паскаль, считать его «русским Паскалем» нельзя — это язык с более простой и естественной для новичка синтаксической структурой. Упрощения проведены, в первую очередь, по методическим соображениям и нацелены на то, чтобы ученик мог быстро и эффективно освоить базовые понятия программирования, не отвлекаясь на непринципи-

альные технические детали. Например, в алгоритмическом языке, в отличие от Паскаля, не нужно думать о том, где ставить и где не ставить точку с запятой, где нужно, а где не нужно использовать операторные скобки begin-end и т.д. КуМир позволяет избавиться от затрат времени на освоение подобных непринципиальных деталей и потратить высвободившиеся драгоценные учебные часы на более содержательные вопросы.

Задания, требующие программирования, могут выдаваться ученикам в электронной форме в виде практикумов со встроенной оффлайн проверкой каждого задания и возможностью сдачи выполненных заданий онлайн. Тем самым ученик сможет в процессе выполнения каждого задания получать отклик от проверяющей программы и отлаживать решение, а учитель будет избавлен и от рутинной проверки правильности составленных учениками программ и от ручного ведения учета результатов каждого ученика. Для этого необходимо предоставить в распоряжение учителя готовые практикумы по всем задачам курса. Ряд практикумов по решению задач в КуМире с автоматизированной проверкой уже разработан К. Поляковым и Д. Кириенко и может быть найден в интернете и скачан для некоммерческого использования. К лету 2017 года авторы планируют разместить в интернете свободно распространяемые практикумы, разработанные для поддержки учебников.

5. Независимость от прикладного ПО

В соответствии с государственным образовательным стандартом учебники содержат разделы, посвященные современной компьютерной технике и программному обеспечению. Рассматриваются, в частности, системы работы с текстами, с графикой, отдельные главы посвящены электронным таблицам и базам данных.

При этом везде соблюдается принцип независимости от конкретного программного обеспечения. Система КуМир оказывается единственным явно используемым программным продуктом. Все остальные программные системы могут упоминаться в качестве примера, но нигде не рассматриваются подробно, ничего не говорится о том, какой пункт меню выбрать, какую клавишу нажать и т.д.

В учебнике излагаются только общие принципы. Например, в главе об электронных таблицах рассматриваются принципы организации таблиц, программирование формул, решаются достаточно сложные

задачи, но все изложение организовано так, что его одинаково легко применить и в Microsoft Excel и в офисных системах СПО. Авторы стремятся научить общим принципам решения задач, а конкретную систему школьники должны осваивать самостоятельно или под руководством учителя.

Леонов А. Г., Райко М. В., Бесшапошников Н. О., Ерёмин Д. Б.
Москва, МГУ, ФГУ ФНЦ НИИСИ РАН, МПГУ

Проект: Мирера `http://www.mirera.ru`

Мирера — система поддержки непрерываемого образования

Объем знаний и умений, который нужно освоить современному школьнику и студенту, давно исчерпал возможность изучения в аудиторные часы учебных заведений, отведенные на соответствующие предметы в процессе образования. Все больше и больше времени требуется ученику для самостоятельных занятий для успешного освоения материала. Однако, вне пределов учебного заведения, школьник и студент не только фактически остаются в одиночестве при выполнении самостоятельных заданий, но и находится в постоянном информационном давлении со стороны социальных сетей, игр и пр., что отвлекает от выполнения учебной нарузки. Поэтому одним из основных методих современного образования должен стать ***непрерываемый*** процесс обучения при котором обучаемому доступны не только онлайн-материалы осваиваемых курсов, включая автоматизированные практикумы, но и сам учитель в режиме 24x7. Площадкой для общения с учениками должны стать социальные сети, в настоящее время занимающие львиную долю времени проводимого в интеренете современными молодыми людьми.

Естественно, что при этом, нагрузка на преподавателя возрастает лавинооообразно. В помощь педагогу и была разработана система (и портал) Мирера (http://www.mirera.ru).

Основной единицей в системе Мирера является курс, который содержит материалы, задания и контольные работы. Обучаемый может выбрать (а иногда и должен) курс и записаться на него. Для студент-

ческих групп педагог сначала создает авторскую группу В контакте[1], члены которой автоматически получают информационные материалы, приглашения на контрольные работы и пр., заранее сформированные педагогом и «привязанные по времени» к текущему курсу. Таким образом педагог избавлен от лишней работы по рассылке материалов и планированию повторяющихся курсов. Материалы и объявления доставляются в группу В контакте в нужный день и время, что повышает активность обучаемых. А освободившееся время учитель может посвятить консультациям и ответам на вопросы,общаясь в группе В контакте со своими учениками.

Обучаемый может выполнять задания в любое удобное ему время, требует лишь доступ в интернет и компьютер/планшет/смартфон с современным браузером.

Естественно, что в Мирера включена автоматизированная система проверки результатов заданий и контрольных. Дополнительно проверяющая система контролирует «списывание» учащимися, используя простейшие алгоритмы сверки сдаваемых работ. Это позволяет исключить до 80-85% плагиата при выполнении заданий.

При разработке проверяющей системы учитывались следующие требования:

- Универсальность для различных языков программирования
- Простота настройки заданий стандартной конфигурации
- Возможность точной настройки каждой ступени проверки
- Возможность использования случайных тестовых данных
- Режим проведения контрольных
- Распределение задач по вариантам
- Легкость просмотра результатов

В настоящее время система Мирера работает в режиме опытной эксплуатации на механико-математическом факультете МГУ им. М. В. Ломоносова в осеннем семестре 2016/2017 года в рамках занятий по курсу «Работа на ЭВМ и программирование» со студентами 1 и 2 курса (языки C и C++) и в Московском педагогическом государственном университете на физичском факультет в рамках выравнивающего курса по программированию (КуМир и ПиктоМир)[3].

В весеннем семестре в систему включены следующие темы:

[1] `http://vk.com`

- Практикум по машинным языкам [2]
- Изучаем OpenGL
- Программрование параллельных процессов (Qt)
- Регулярные выражения

В краткосрочной перспективе планируется добавить возможность создания собственных компиляторов специально для курсов.

В соответсвии с требованиями ФГОС на портале Мирера также можно формировать портфель достижений учеников, в который, по желанию преподавателя, могут входить выборки выборки выполнненых заданий, результаты прохождения курсов, включая сертификаты о прохождении курсов и олимпиад, индивидуальные работы ученика, с приложениями самих работ: текстов, бумажных или электронных документов, фотографий и т.д. [1]

На основании результатов, собранных в портфеле достижений, не только педагог может формаировать общее предствление о результатах деятельности ученика, но и его родители (опекуны) также имеют досуп к порталу и могут знакомиться с успехами своего ребенка.

Литература

[1] *Леонов А. Г. Первин Ю. А.* Качественные оценки эффективности методики обучения элементам информати-ки в пропедевтическом курсе// Ярославский педагогический вестник — 2015 — № 5

[2] *Леонов А. Г. Прилипко А. А.* Автоматизированный практикум по машинным языкам — основа изучения языков программирования// Сборник научных статей: Интеграция отечественной науки в мировую: структурные преобразования и перспективные направления развития. Россия, Санкт-Петербург — 2016.

[3] *Леонов А. Г. Прилипко А. А.* Разработка и внедрение компьютерных практикумов в учебные курсы программирования в школе и вузе // Сборник научных статей по итогам международной научно-практической конференции. Россия, Санкт-Петербург — 2015, стр.116–120

Воронин Игорь Вадимович, Воронина Вероника Вадимовна
г. Шатура, г. Павлово-на-Оке, ИПЛИТ РАН, МБОУ СОШ № 7

Проект: УМКИ `http://www.umkikit.ru/`, `http://umki.vinforika.ru/`

Образовательная робототехника УМКИ на основе АЛЬТ в Зимней школе

Аннотация

Обсуждается опыт проведения зимней школы ФИЗТЕХ-Потенциал для учеников старших классов, как изучение свойств физических объектов и сравнении их с виртуальными. Учащиеся проводили физические эксперименты, загружали наборы «сырых» данных от Интернет Вещей, обрабатывали и получали координаты перемещения мобильных роботов

Зимние каникулы наступившего 2017 года, школьники могли провести их по разному: кто-то катался на лыжах и коньках, кто-то играл в компьютерные игры, или лежал на диване перед телевизором.

Для ребят, выбравших Зимнюю школу «Физтех — Потенциал» занятия были организованы следующем образом: первая пара — математика, вторая физика а третья информатика-робототехника.

Занятия по информатике-робототехнике, были ориентированы на учеников 9, 10 и 11 классов. Для курса была выбрана авторская программа В. Ворониной `http://umki.vinforika.ru/`

Курс начался с обзорной лекции включающей видео ролики о различных типах наземных, воздушных и подводных роботов, о способах их передвижения, и протоколах связи, которые используются для обмена телематическими данными.

После этого, с большим удовольствием, ребята просто погоняли роботов между препятствиями. При этом они узнали разницу между дискретным и непрерывным режимами управления роботами.

Дискретный режим — когда всякое передвижение робота фиксируется датчиком движения — энкодером и робот проезжает строго то расстояние, которое ему задано программой. А непрерывный, действует до тех пор, пока нажата клавиша на пульте управления.

Ребята зафиксировали в лабораторной тетради связь между импульсами от энкодера с дистанцией пройденного пути.

На второй день ребята вспомнили основы логики из школьного курса информатики и сами собрали из радио конструктора простейшие автоматы работающие на основе булевой алгебры.

Следующее занятие с платами ардуино, предусматривало серию лабораторных опытов начиная с базовых тестовых примеров, таких как управление миганием светодиода и заканчивая построением прибора фиксирующего расстояние от платы до препятствия в соответствии с данными получаемыми от ультразвукового датчика. Теоретическая часть предусматривала знакомство с такими понятиями как ШИМ, как при помощи такой модуляции можно изменять окраску многоцветного светодиода с очень красивыми переливами. Участники Зимней школы научились пользоваться средой ArduinoIDE и разработали свои первые программы на языке Си.

Далее, ребята продолжили знакомство с датчиками — собрали свой собственный энкодер: оптический или магнитный, по типу датчика Холла. Показания с энкодеров научились отправлять по радиопротоколу, для расчета скорости вращения колес робота.

В дальнейших экспериментах участникам пришлось вспомнить физику и по второму закону Ньютона ребята рассчитывали тягу электродвигателей, путем использования результатов автоматизированных замеров времени и длины пути из LOG-файла, который каждая группа сгенерила самостоятельно.

Поскольку реальные объекты отличаются от виртуальных, то следующим экспериментом было измерение угла поворота тележки робота в зависимости от заданных величин датчиков энкодеров. Работа проводилась в команде, когда один участник группы задавал энкодрами нужный диапазон величин и отправлял команду по радио эфиру на машинку, второй отмечал маркером пройденные точки пути и строил пересекающиеся прямые, а третий делал замеры углов и выполнял вычисления. Данные от робота поступали в шестнадцатеричной форме и для того чтобы правильно построить график было необходимо преобразовать их в десятичный формат. Полученные и обработанные результаты сводились в табличку, на основе которой делались выводы о различиях между виртуальным и реальным роботами.

На четвертый день приступили к определению местоположения робота в пространстве, для чего строили виртуальные и реальные сетки координат, вычисляли ожидаемое место положение робота, а затем экспериментально отмечали реальное местонахождение на плоскости. И на основе полученных данных определяли погрешности. На буду-

щее планируется осуществлять загрузку данных из лог файла в базу данных и путем сравнения значений в различных таблицах проводить анализ больших объемов данных, знакомясь с азами технологии data mining.

В завершающий день — участники зимней школы говорили о бизнесе, обсудили чем отличается фундаментальная наука от инновационной деятельности, обдумали как запустить свой стартап. После обзорной лекции про конечные автоматы ребята создали самостоятельно программу на языке Питон для обработки «сырых» данных полученных из ЛОГ файла, и по этим обработанным данным смогли создать облако точек с координатами перемещения робота по рабочему полю. Тем самым, ребята самостоятельно решали высокопрофессиональную задачу локализации — определения местоположения мобильного робота в заданном пространстве.

Дипломной работой каждого участника было оформление своего проекта, который можно показать в школе и использовать как заготовку для дальнейшего представления на научно-практической конференции.

Программное обеспечение было целиком свободно распространяемое ПО. В данном случае использовалась ОС ALT Linux, среда разработки Arduino IDE, Geany IDE и программа для управления роботами smartcar, скачанная с сайта `http://umkikit.ru`, она может быть использована совершено бесплатно для любой операционной системы Linux, Windows, MacOs. Платы R5 с Arduino Nano, набором датчиков и светодиодов с уже распаянными сопротивлениями, были закуплены в магазине `http://lartmaster.ru/`.

Сами роботы связывались между собой по протоколу Zigbee. Для связи их с ПК использовались USB адаптеры. И роботы и USB адаптеры были закуплены через интернет магазин `http://umkikit.ru/`

Евгений Синельников
Саратов, ООО «Базальт СПО»

Проект: Arduino templates for Robots https://github.com/centrit/AXRobot

Анализ свободных средств разработки для образовательной робототехники

Аннотация

Современные робототехнические конструкторы требуют специальные средства разработки. Подобные средства разработки позволяют снизить порог вхождения при программировании автономных робототехнических устройств, оснащённых датчиками и исполнительными механизмами, то есть роботов. На текущий момент возможности свободного программного обеспечения в области образовательной робототехники ограничены, а их разработка имеет свою актуальную проблематику. В данном докладе представлен анализ возможностей и ограничений средств разработки для программирования роботов на примере подготовки олимпиадных задач и участия в олимпиадах по робототехнике.

Задачи образовательной робототехники

Ответ на вопрос: «Что такое робот?», — с точки зрения содержания и методики преподавания, можно свести к двум вариантам:

- Робот — это автономное, программируемое, программно-аппаратное решение с набором датчиков и исполнительных механизмов;
- Робот — это управляемый набор программно-аппаратных компонент, осуществляющий различные производственные и иные операции.

В первом случае, основной упор делается на разработку и программирование автономной робототехнической модели, например, для решения олимпиадных задач, во втором — на подробное изучение функциональных возможностей отдельных программно-аппаратных компонент, представленное в иностранных образовательных заведениях, как STEM-направление [8].

Программно-аппаратные средства

Данный доклад посвящён разбору возможностей и ограничений средств разработки для программирования роботов в рамках первого подхода. При этом, одну из важнейших функций играет поддержка средой разработки возможности программирования той или иной робототехнической платформы через различные проводные и беспроводные интерфейсы. Стандартом де-факто, в качестве такой платформы в задачах олимпиадного, робототехнического программирования, на текущий момент, является конструктор Lego Mindstorms EV3 [1]. В качестве свободной, недорогой альтернативы ему представлены различные решения на базе платформы Arduino [3], которая включает в себя открытую аппаратную платформу для различных типов микроконтроллеров и среду разработки, позволяющую выполнять прошивку этих микроконтроллеров через стандартный USB-интерфейс.

В силу открытости платформы Arduino крупные компании самостоятельно реализуют для неё поддержку своих отладочных плат — Edisson [6] от Intel, LaunchPad [7] от Texas Instruments, ESP8266 [4] от Espressif и др. Кроме того, для большинства недорогих датчиков и исполнительных механизмов, как правило, доступно множество примеров исходного кода.

Таким образом, одной из основный свободных альтернатив, в качестве аппаратной базы для образовательной робототехники, является именно платформа Arduino. Тем не менее, реальное применение Arduino, в качестве замены для весьма дорогостоящих Lego Mindstorms, в задачах образовательной и олимпиадной робототехники до сих пор не осуществлена, в силу следующих причин:

- основной язык программирования для Arduino — C++;
- целостная сборка готового решения на базе доступных примеров, зачастую, требует реализации элементов операционной системы реального времени;
- различные аппаратные средства требуют подключения и монтажа электронных компонент через различные аппаратные интерфейсы (I2C, UART, SPI), а также реализацию протоколов для проводных и беспроводных интерфейсов (Ethernet, Wifi, Bluetooth и др.);

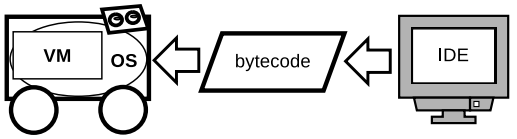
- Отсутствуют и требуют дополнительного программирования кнопки управления, модуль визуального отображения и вывода звука;
- Требуется отдельное подключение исполнительных механизмов через специальные силовые драйверы с повышенным напряжением, а также подключение и программирование энкодеров.

С одной стороны, такой минимализм позволяет максимально подробно изучить дополнительные модули, аппаратные интерфейсы и спецификации на протоколы, а также низкоуровневое программирование с использованием портов ввода-вывода, прерываний, аналого-цифровых преобразователей и таймеров. Но это актуально, скорее, в рамках демонстрации выполнения операций осуществляемых роботом. Готовое, автономное решение для олимпиадного соревнования в обозримые сроки, силами школьников подготовить таким образом довольно сложно.

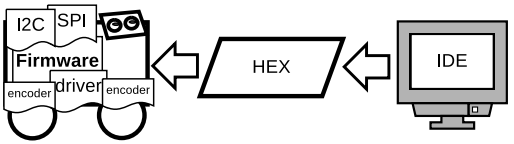
Средства разработки

Тем не менее, в качестве базовой, свободной среды разработки для доступных по стоимости робототехнических решений являются средства платформы Arduino.

При этом, например в Mindstorm, где все аппаратные модули заранее реализованы, за программную часть управления аппаратными средствами отвечает операционная система на базе Linux, базовая прошивка, включающая специализированную виртуальную машину и генератор байткода для этой виртуальной машины, реализованный в рамках пропритарной графической среды разработки, работающей под MacOS и Windows (есть вариант запустить с помощью wine).



В качестве альтернативы для программирования Mindstorm существует ряд дополнительных свободных проектов, загружаемых с внешней SD-карты:



- EV3Dev[5] сборка на базе Debian (не имеет среды разработки, позволяет использовать любые языки программирования доступные в прошивке, при подключении через SSH — shell, python, C/C++, NodeJS и др.);
- LejOS на базе Java (имеет плагин для среды Eclipse);
- MonoBrick на базе .Net/Mono (имеет плагин для среды Xamarin Studio);

Кроме несовместимых с базовой прошивкой свободных альтернатив существует проприетарный набор продуктов RobotC, который самостоятельно генерирует байтокод для виртуальной машины из базовой прошивки, поддерживает дополнительные платформы, в частности Arduino, а также предоставляет трехмерную среду-эмулятор для отладки программ.

Подобное решение представлено в свободной, кроссплатформенной, графической среде разработки TrikStudio (первичное название qreal). Данная среда поддерживает, как собственную платформу TRIK, так и платформу Mindstorm. При этом поддержка платформы Arduino отсутствует, но имеется встроенная, двухмерная среда-эмулятор для отладки программ.

Одним из перспективных вариантов реализации универсальной, графической среды разработки является следующее поколение программной платформы ScratchX (или Scratch 2.0), выполненное на базе web-технологий (также доступная в offline режиме через технологию Adobe Air). В этой программной платформе поддержка различных аппаратных платформ реализована в виде плагинов для браузера Chrome. К сожалению, на текущий момент, поддержка Linux в этом решении отсутствует.

При этом, для платформы Arduino имеется имеется брошенный, не развивающийся, но пока работоспособный, проект графического расширения Ardublock, выполненный в виде java-плагина, реализованный на базе OpenBlocks в виде таких же блоков, как в проекте

Scratch. На текущий момент, это наиболее простое, доступное средство графического программирования для Arduino.

Кроме десктопных решений, для платформы Arduino реализован ряд web-эмуляторов с поддержкой различного оборудования. Наиболее популярным среди них является набор 123D/circuits.io от Autodesk. С недавнего времени, в рамках проекта Arduino, реализовано собственное web-решение Arduino Create [2] с возможностью прошивки контроллеров через плагин для браузера.

Возможности и ограничения в олимпиадных задачах

Практика подготовки к решению олимпиадных задач по робототехнике даёт ряд противоречивых ограничений — сложность освоения среды разработки (включая язык программирования) и необходимость реализации не тривиальных алгоритмов (включая работу со структурами данных). В первом случае графическая среда разработки помогает снизить порог вхождения, во втором — существенно замедляет или, вообще, не позволяет достаточно выразительно реализовывать алгоритмы.

Создание цепочки функций со входными и выходными параметрами, работа с массивами, списками, стековыми структурами, а также строчные операции в графических средах зачастую выглядят громоздко и «нечитабельно».

В связи с этим возможности платформы Arduino выглядят достаточно перспективно в случае генерации настроек и базового кода на основе шаблонов. Особенно для средств разработки и симуляции, доступных в виде web-решений. Тем не менее, актуальность генерации кода на основе визуальных шаблонов, диаграмм и блок-схем сохраняется для олимпиадных и образовательных задач по робототехнике. Отсутствие же таких средств, резко снижает возможности платформы Arduino по сравнению с платформой Mindstorm.

На текущий момент существует целый ряд возможностей и вариантов реализовать двухступенчатую схему программирования (как в графическом, так и в текстовом виде) в рамках платформы Arduino. Возможно, реализацию такого решения имеет смысл выполнить в виде web-приложения.

Литература

[1] *Л. Ю. Овсяницкая, Д. Н. Овсяницкий, А. Д. Овсяницкий*, Курс программирования робота EV3 в среде Lego Mindstorms EV3 / М. Издательство «Перо», 2016

[2] *Alice Pintus*, Sneak peek on the new, web-based Arduino Create, 2015, `https://blog.arduino.cc/2015/05/05/sneak-peak-arduino-create/`

[3] Arduino, Getting Started with Arduino and Genuino products, 2016, `https://www.arduino.cc/en/Guide/HomePage`

[4] *Neil Kolban*, Kolban's Book on the ESP32 & ESP8266 / Leanpub, 2016

[5] *Ralph Hempel and David Lechner*, Getting Started with ev3dev, 2016, `http://www.ev3dev.org/docs/getting-started/`

[6] *Stephanie Moyerman*, Getting Started with Intel EdisonSensors, Actuators, Bluetooth, and Wi-Fi on the Tiny Atom-Powered Linux Module / Maker Media, Inc, 2015

[7] *Texas Instruments*, Getting Started with Energia, 2016, `http://energia.nu/guide/`

[8] *U.S. Department of Education, Science, Technology, Engineering and Math*: Education for Global Leadership, 2015, `https://www.ed.gov/stem`

Дмитрий Костюк, Олег Латий, Анастасия Маркина
Брест, Брестский государственный технический университет

Проект: UXDump `https://bitbucket.org/AsyaAliset/uxdump`

Клавиатура как соавтор: биометрическая оценка качества набора текста на сенсорном экране

Аннотация

В ходе проверки предположения о том, что экранные клавиатуры с функцией эвристики являются первой за долгое время успешной инновацией в сфере клавиатурного ввода, выполнено сравнение эффективности набора текста с помощью стандартной аппаратной клавиатуры и двух клавиатур с открытым кодом: onboard из Ubuntu Linux 16.04 и AOSP keyboard из Android 6.0.1. Тестирование проведено для текстов трёх градаций сложности. Представлены результаты биометрической оценки когнитивной и физической нагрузки пользователей в процессе

набора, рассмотрены критерии оценки. Обсуждаются различия в эффективности ввода традиционных экранных клавиатур и современной клавиатуры, использующей динамический словарь и эвристику для исправления промахов в пользовательском наборе, их возможное влияние на рыночную долю соответствующих операционных систем.

Введение

Исторически, компьютерная клавиатура является наименее развивающимся периферийным устройством. Все изменения, которые она претерпела за последние десятилетия, сводятся к незначительным изменениям формы и количества клавиш. Проекты по созданию «новой клавиатуры» оказываются неинтуитивны и требуют длительного обучения, либо имеют цену, несравнимую с достигнутыми улучшениями (пример — клавиатуры А. Лебедева с дисплеем в каждой клавише, стоившие ок. $1000). В результате, альтернативные аппаратные клавиатуры нашли применение в узкой области — для использования людьми со слабо-выраженными нарушениями моторики рук (например, туннельным синдромом запястий).

Концепция получила новое развитие с ростом популярности емкостных сенсорных дисплеев. Экранные клавиатуры вызывали много нареканий в отношении скорости набора и количества ошибок из-за нестандартных размеров и отсутствия тактильной связи, что простимулировало превращение клавиатуры в «соавтора», занимающегося исправлением и дополнением набираемого текста. Использующие данный подход мобильные системы (Android, iOS) получили преимущество перед системами, рассматривающими клавиатуру в традиционном качестве (MS Windows, дистрибутивы GNU/Linux), и по нашему мнению это может оказаться одним из ключевых факторов, отвечающих за ничтожную долю традиционных Linux-систем в соответствующем сегменте. Выяснению количественной разницы в эффективности ввода между традиционными экранными клавиатурами и современной экранной клавиатурой, использующей динамический словарь и эвристику для исправления промахов в пользовательском наборе, и посвящено настоящее исследование.

Выбор тестовых заданий

Для исследования были выбраны 3 варианта текстов, в порядке уменьшения сложности Q ($Q_1 > Q_2 > Q_3$):

- Текст на заведомо неизвестном пользователю европейском языке (Q_1) — фрагмент на псевдо-латыни, т. н. «Lorem Ipsum», используемый в издательском деле при тестировании шрифтов (фактически, перемешанные фразы из трактата Марка Туллия Цицерона «О пределах добра и зла»).
- Сложный текст на родном языке (Q_2) — фрагменты описаний природы в русской литературе (использованы «Война и мир» Л.Н. Толстого, «Леди Макбет Мценского уезда» Н.С. Лескова и «Апрекарь» В.Н. Орлова).
- Текст, характерный для обмена мгновенными сообщениями (Q_3) — верлибры Ч. Буковски из сборника «Стихи последней ночи на земле» (отсутствие рифмы, слабо выраженная пунктуация, предельно простой стиль и другие особенности текста делают данный материал качественной натурной моделью интернет-диалога).

Для тестирования были выбраны (рис. 1) референсная аппаратная клавиатура (а) и две экранные клавиатуры, сохраняющие визуальное сходство с аппаратной: onboard из дистрибутива Ubuntu Linux 16.04 в роли классической экранной клавиатуры (б) и Android Open Source Project Keyboard с автоматическим исправлением набора и строкой подсказок автодополнения (в).

В тестировании участвовали 22 студента БрГТУ дневного и вечернего отделений в возрасте 17–39 лет. Ввод текста выполнялся на ноутбуках Asus и близких по размеру планшетах с Ubuntu либо Android x86. При проведении опытов использована система тестирования UXDump [1].

Критерии оценки и результаты

Исходно рассматривались 4 группы показателей эффективности: длительность выполнения заданных действий τ, число допущенных ошибок e, а также 2 биометрических показателя: частота сердечных сокращений (ЧСС) p и концентрация внимания оператора β. Выбор исходных критериев и общей методики тестирования является автор-

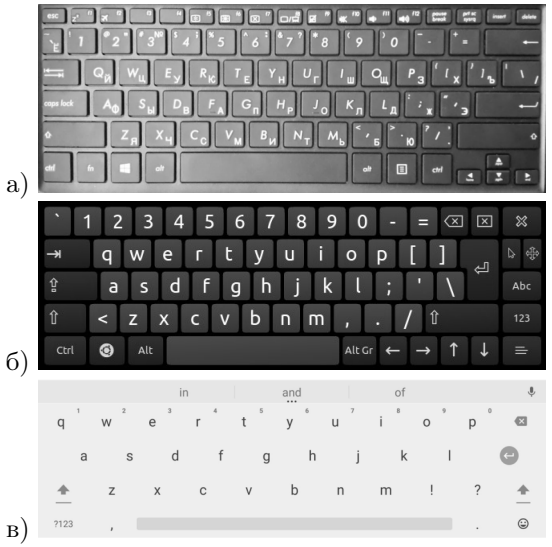


Рис. 1: Клавиатуры, участвовавшие в тестировании

ским и успел хорошо зарекомендовать себя в ряде предыдущих работ [2].

По факту подсчитывались только ошибки e_1, пропущенные оператором при наборе (т. к. исправленные ошибки e_2 вносят вклад в τ). В качестве показателя скорости выбран темп набора текста ν, равный числу правильно набранных символов, генерируемых оператором за секунду, а для оценки ЧСС и концентрации внимания — среднее значение сердечного ритма за время выполнения теста $< p >$ и среднее значение концентрации внимания $< \beta >$. В роли последнего использована метрика «Attention» энцефалографа Neurosky Mindwave, связанная с β-ритмом головного мозга [3, 4], а для мониторинга ЧСС применялись модули снятия данных с фитнес-трекеров [1].

Оценка результатов тестирования без деления по сложности текста представлена в таблице 1. Аппаратная клавиатура предсказуемо оказалась наиболее быстрым средством ввода (максимум ν) и наиболее интуитивным (минимум β). Однако физическая нагрузка и процент ошибок в среднем меньше при печати на клавиатуре Android (минимальные средние значения p и e_1). Клавиатура Ubuntu демон-

	ν, симв./сек	β, о.е.	p, уд./мин	e_1, %
Аппаратная клавиатура	1,44	48,10	73,26	0,41
Клавиатура Ubuntu	0,90	49,44	79,09	0,46
Клавиатура Android	0,92	50,62	73,11	0,29

Таблица 1: Сравнение эффективности набора текста

стрирует худшую скорость, максимальные среднюю ЧСС и число ошибок, среднюю концентрацию внимания (что свидетельствует о большем уровне стресса).

Учет сложности текста дает менее однозначную картину. Максимальный темп зарегистрирован на аппаратной клавиатуре для всех текстов, а минимальный — на клавиатуре Ubuntu для текстов сложности Q_1 и Q_3, в то время, как набор текстов Q_2 оказался наиболее медленным при использовании клавиатуры Android (видимо, полнота словаря Google сильно уступает словарному запасу русских писателей). Также русская литература показала минимальные значения ЧСС для аппаратной клавиатуры (инспирированные размеренным ритмом описаний природы). ЧСС клавиатуры Ubuntu оказалась пропорциональна сложности текста, а ЧСС при использовании клавиатуры Android продемонстрировала обратную зависимость (в связи с особенностью обработки человеком автозамены; тем не менее, положительный эффект автозамены превалирует, т. к. темп набора прямо пропорционален сложности текста для всех экранных клавиатур).

Концентрация внимания для аппаратной клавиатуры и клавиатуры Android имеет схожую динамику (максимум на Q_2); при этом текст неизвестного языка требовал большего сосредоточения при работе на аппаратной клавиатуре и меньшего — на клавиатуре Android, благодаря успешным автозамене и автодополнению (из-за латинской основы многих европейских языков хорошо «работала» эвритстика).

Величина e_1 обратно пропорциональна сложности текста Q для аппаратной клавиатуры и клавиатуры Ubuntu, в то время как автозамена клавиатуры Andoid снова не справляется со словарным запасом Q_2, творчески приумножая ошибки.

Т. о., хотя степень автоматизма, достигаемая при работе с аппаратной клавиатурой, остаётся непревзойдённой, экранная клавиатура в роли активного агента повышает скорость и качество набора простых и умеренно-сложных текстов (и только в действительно слож-

ных случаях является источником стресса, ошибок и снижения эффективности). Классическая же экранная клавиатура почти всегда оказывается существенным негативным фактором при внедрении использующих её платформ.

Литература

[1] *Маркина А.* Система параллельного тестирования эффективности человеко-машинного взаимодействия // Тринадцатая конференция разработчиков свободных программ: Тезисы докладов / Калуга, 01-02 октября 2016 г. М.: Базальт СПО, 2016. – С. 32–37.

[2] *Костюк Д., Дереченник С., Шитиков А.* Оценка эффективности управления окнами в современных графических оболочках // Седьмая конференция «Свободное программное обеспечение высшей школе»: Тезисы докладов. – Переславль, 28–29 января 2012 года. М.: Альт Линукс, 2012. – С. 20–23.

[3] *Dhali S.* A Study of Brainwave eSensing Activity. Department of Computer Science, Malmo University (electronic publication). `https://www.overleaf.com/articles/bci/mcsvkjwhcffb/viewer.pdf`

[4] *Sezer A., Inel Y., Seçkin A.Ç., Uluçinar U.* An Investigation of University Students' Attention Levels inReal Classroom Settings with NeuroSky's MindWave Mobile (EEG) Device. // Proc. of IETC 2015 int. conf. , May 27–29, Istanbul, Turkey. – p. 88–101.

Татьяна Сундукова
Тула, Тульский государственный педагогический университет им. Л. Н. Толстого

Основные элементы геймификации в LMS Moodle

Аннотация

В докладе рассматриваются основные элементы геймификации, такие как очки, бейджи, доска почёта, уровни, квесты, миссии и приводятся инструменты Moodle для их реализации.

В настоящее время одним из актуальных направлений развития образовательных технологий является геймификация. Геймификация — это использование подходов, которые свойственны компьютерным

играм в так называемых неигровых контекстах с целью привлечения пользователей, повышения их вовлеченности в решение задач, использования продуктов и услуг. Геймификация в обучении выражается в специально сконструированной, на основе игровых элементов и игрового дизайна, оболочке для образовательного процесса.

Перечислим популярные игровые механики (элементы геймификации), которые разработчики учебных проектов могут использовать в LMS Moodle.

Очки. Очки являются мерилом прогресса, базовым элементом геймификации. Они присуждаются учащимся за выполнение каких-либо заданий или иные виды активности, предусмотренные учебным процессом. Они помогают мотивировать учащихся на протяжении всего учебного курса и/или занятий. Очки могут использоваться для управления определенными действиями учащихся, обозначать статус в курсе, и даже предоставлять награды или возможности для доступа к новые уровням обучения. При этом само начисление очков за определенные достижения, без выстраивания статусных отношений между учащимися, не приводит к геймификации образовательного процесса. Реализация в Moodle посредством: журнала оценок, «Grade points».

Бейджи (награды, значки, медали, похвала, сертификаты, призы и другие). Они являются дополнительным вознаграждением за особые успехи, которые определяет разработчик учебного курса. Это своего рода знаки отличия, получаемые обучающимися за конкретные действия и достижения. Можно награждать учащегося за быстро или качественно выполненную работу, а также награды за уникальные идеи, командную работу. Выбор «повода для награды» зависит от конкретных целей обучения, это может быть цель сделать какое-то количество дополнительных заданий, а также скорость и качество решения определенных задач. Чаще всего такие бейджи снабжаются комментариями вида «самый активный ученик», «за лучшую работу», «за лучшую идею» и т.п. Бейджи помогают мотивировать учащихся сделать больше, качественнее, быстрее, внедрять инновации, искать новые решения и работать наиболее эффективнее, чем другие. Такие системы поощрения побуждают учащихся к получению новых знаний и раскрытию собственных возможностей. Кроме этого, бейджи — это дополнительная обратная связь, которая демонстрирует учащемуся, что он на верном пути, развивает и совершенствует свои умения и навыки. Реализация в Moodle посредством: «Stamp collection», «Open

Badges», «Custom certificate», « Gold, Silver, and Bronze (GSB) Medals» и т.п.

Доска почёта (рейтинг лидеров). Это таблицы и графики, отражающие рейтинг (ранжирование) учащихся и их позицию по отношению к другим участникам образовательного процесса. Они помогают учащимся оценить свою эффективность в сравнении с результатами других учащихся. Рейтинги всегда мотивируют пользователей к более активным действиям. Поощряйте конкуренцию между ними. Но данный элемент должен быть использован с осторожностью, поскольку сравнение учащихся между собой может иметь как мотивирующий, так и демотивирующий эффект. Для того, чтобы свести к минимуму возможную демотивацию учащихся, можно сделать доски почёта по разным параметрам или промежуточные доски, например, раз в неделю. Реализация в Moodle посредством: «Ranking block», «Badge Ladder».

Уровни. В первую очередь, это показатель продвижения к цели. В системе образования – это может быть номер курса (класса), а при изучении конкретной дисциплины — модули, разделы, темы, задания и т.п., которые можно представить в виде уровней игры, добавив постепенное усложнение процесса. Уровни помогают учащимся понять, когда они достигнут того или иного рубежа или окончательной цели, а также они показывают обучающимся их позицию относительно других участников по количеству заработанных в процессе обучения поощрительных элементов. Многие используют уровни как возможность обеспечить дополнительную обратную связь, а также определить области для улучшения. Реализация в Moodle посредством: «Levelup!», «Level availability», ограничения доступа, разграничения по потокам.

Квесты, миссии. Они представляют собой задания, предполагающие наличие сюжета, которые учащиеся выполняют в ходе игры. Квесты и миссии придают игровому процессу структуру, ведет учащегося по сюжету игры. В учебных курсах, которые состоят из множества небольших повторяющихся заданий, есть возможность составления квестов из разного набора однотипных заданий. Реализация в Moodle посредством: «Game», «Exabis Games» и т.п.

В заключение отметим, что LMS Moodle в полной мере приспособлена для реализации игровых механик в учебном процессе. Благодаря, разработке функциональным дополнениям и модулей, процесс

обучение можно организовать интересно, эффективно, конкурентоспособно и увлекательно.

Михаил Шигорин
Москва, Базальт СПО
http://altlinux.org/ports/e2k

Альт на «Эльбрусе»: 2017

Аннотация

После создания полноценной сборочной среды на рабочей станции «Эльбрус-401» пошёл рабочий процесс собственно портирования пакетной базы ALT; в докладе разбираются обнаруженные узкие места, способы их преодоления и полученные на данный момент результаты.

The routine process to actually port ALT package base to e2k architecture has started out upon completion of the automated build environment. This report looks at the bottlenecks found, methods used to overcome those, and current results.

Мы уже рассказывали о проекте портирования репозитория Альт на отечественную архитектуру «Эльбрус», а точнее, на процессор «Эльбрус-4С»; по состоянию на момент, описанный в предыдущем докладе, была достигнута точка работоспособности автоматической сборочной системы hasher (и далее до примерно 250 srpms), но вскоре была достигнута и точка преткновения в виде проблемы между генератором зависимостей в составе rpm-build (cpp.req) и особенностями компилятора lcc (вывод препроцессора несколько отличается от gcc).

Таким образом, весна и лето по сути выпали из работ по этой теме: можно было заниматься чем-либо, не затрагивающим libX11, но это потребовало бы существенных заведомо лишних усилий по добавлению «ручек» сборки без x11 и заметно ограничивало возможные результаты (не говоря уже о том, что неполные сгенерированные зависимости могли скрытно образоваться и в других пакетах). Репозиторий замер на полутысяче srpms.

Осенью вдруг объявился сам автор cpp.req, Алексей Турбин, и на материале почтового обсуждения с примерами вывода обоих препроцессоров с первого раза написал заработавший патч[4].

После сборки базовых графических библиотек процесс портирования обрёл второе дыхание — последовали openldap, python3, cups,

cmake, а за ними «потянулись» требующие их для сборки пакеты. Ускорению способствовало и получение новой версии lcc 1.21 со значительно улучшенной совместимостью с более новыми версиями gcc и стандартов.

Вскоре определилась и ставшая достижимой ближайшая цель: минимальная графическая система на базе Xfce, перекликающаяся с дистрибутивом Альт Образование.

По состоянию на середину января 2017 года собрано более 800 исходных пакетов, при этом наработанный репозиторий уже позволяет собирать достаточно многие пакеты без каких бы то ни было правок, разве что порой вместо добавления «ручек» отбирая версии из предыдущих стабильных веток (нередко мешают завязки на systemd). Работает сборка архивов чрутов при помощи mkimage-profiles. Реализована параллельная пересборка репозитория, занимающая на машине «Эльбрус-401» уже около суток. Потихоньку возвращаются в положение «вкл» выключенные было «ручки» от udev до gl.

Из оставшихся проблем:

1. doxygen!
2. не собирается текущий perl 5 (даже новым компилятором);
3. часть базовых пакетов (binutils, gmp, strace) остаётся alien-изированной вместо «родной» сборки;
4. также всё сложно с guile18 (т.е. и autogen), требуется полноценный порт реализации сборки мусора;
5. наконец, alien-изированному lcc требуется 32-битная glibc (тоже перепакована).

Часть из них относится к классу «сесть и сделать», а некоторые придётся решать вместе со специалистами МЦСТ (в первую очередь это относится к guile).

Параллельно предполагаются работы по экспериментальному и затем промышленному формированию загрузочных образов операционной системы.

Как мне кажется, выбранный подход с возможно более ранним переходом к сборке пакетов в hasher в «многозадачных» условиях себя оправдал.

Ссылки

1. `http://altlinux.org/ports/e2k`
2. `http://altlinux.org/bootstrap`

3. http://altlinux.org/hasher
4. https://github.com/svpv/altrpm/commit/3d70a2c81e1524b93a20b89a43c214f74acb259b

А.А. Ковалевский, А.Н. Пустыгин
Челябинск, Челябинский государственный университет

Использование системы выделенных признаков для задач поиска по исходному тесту.

Аннотация

В данном исследовании используются инструменты извлечения данных из открытого программного кода в целях поиска по условиям, формирование и формализация которых слишком сложны или невозможны иным способом. С этой целью строится эквивалентное представление исходного текста, являющееся набором неделимых блоков исходного текста. Использование системы выделенных признаков для блоков исходного текста показывает возможность контекстного поиска по синтаксическим признакам.

Задача статического анализа программных систем является одной из традиционных способов улучшения программ [1]. В предшествующих разработках [2] было предложено и реализовано универсальное промежуточное представление (УПП) открытого исходного текста, предназначенное для последующего преобразования, анализа и извлечения информации из исходного кода программ [8]. На основе этого набора данных был построен прототип генератора эквивалентного представления, предназначенных для решения некоторой совокупности задач над исходным текстом.

Алгоритм получения набора данных состоит из двух этапов преобразований. Сначала из УПП путем атрибутирования метками потока данных (операции над данными) и потока управления (вызовы, передача управления) получается эквивалентное представление UIRDCF[7]. На втором этапе последовательно просматривается глоссарий и для каждой его записи выполняется модификация каждого кванта UIRDCF, если он соответствует шаблону признака из глоссария. Полученное эквивалентное представление назовем UDCFM.

Для тестирования использовались проекты с открытым исходным кодом pugixml [3], jsoncpp [4], lz4 [5] и zlib [6].

Была предложена тестовая система признаков (глоссарий), сформированная из эвристических соображений, соответствующих алгоритмическому смыслу описываемого кванта.

Глоссарий признаков описывается в текстовом файле в формате XML документа и разделен на две категории квантов: метод и класс. Каждая категория содержит набор шаблонов, соответствующих искомым признакам квантов, в виде блока условий:

```
<type name="<имя признака>" title="<описание признака>"
                                        [параметры признака] >
    <node name="<имя тега>" [параметры условия] >
        <attr name="<имя атрибута>" [параметры атрибута] />
        ...
    </node>
    ...
</type>
```

Каждый узел условия конкретизирует свойства искомого признака кванта, например в категории «метод» для признака «получатель поля» (getter) заданы следующие условия шаблона:

```
<node name="MethodDecl">
<attr name="const" vb="true" />
<attr name="argc" vi="0"/>
</node>
<node name="FieldRef" tags="return" parent="this" />
```

Что соответствует:

- метод является константным,
- метод не имеет аргументов,
- поле объекта this напрямую возвращается оператором return.

Атрибут tags второго условия определяет набор меток из эквивалентного представления UDCFM, которым должен обладать узел УПП.

Формат шаблонов позволяет записывать условия, требующие семантического анализа, который дополнительно выполняется утилитой на первом этапе обработки запроса. Результат этого анализа записывается в метаданные узлов дерева разбора, к которым относится: принадлежность типу, принадлежность классу определения, узел дерева точки объявления. К метаданным добавляется информация

Стр.	Имя метода
6273	convert_number_to_boolean
6026	tolower_ascii
6332	convert_number_to_string
1386	convert_buffer

Таблица 1: Примеры методов признака «функция-конвертер» (файл pugixml.cpp)

UDCFM, которая включает в себя итеративно дополняемый список уже вычисленных признаков квантов на базе анализа предыдущих шаблонов.

Предложенная система признаков разбивает достаточно крупный проект на некоторое количество уникальных групп квантов, т. е. групп методов или классов. Например, для проекта pugixml с 621 методом количество групп методов составляет 200, а для проекта jsoncpp с 532 методами количество групп методов 101.

Примеры выделенных признаков квантов исходного текста

Для шаблона признака «функция-конвертер» (func-converter) в тестовом проекте pugixml утилита показывает методы, имеющий в названии составляющую «convert» (sic!) или характерный предлог «to» (Таблица 1), что является симптоматичным совпадением, т.к. имена шаблонов не привязаны к именам идентификаторов. Признак «функция-конвертер» полностью соответствует 39 методам проекта и входит в 16 групп строгих дайджестов.

При поиске шаблона признака «проверяющая функция» (func-checker) в тестовом проекте pugixml находится функция «check_string_to_number_ format» (строка 6395, файл pugixml.cpp) с ключевым словом «check» в имени.

При поиске шаблона признака «проверяющий метод» (checker) в тестовом проекте jsoncpp находятся соответствующие методы, в названиях которых присутствует глагол «is» (Таблица 2)

Например, метод Value::isBool в строке 1262 файла json_value.cpp сравнивает значение поля «type_» с константой «booleanValue» и возвращает булевый результат сравнения (Таблица 3).

Стр.	Имя метода
248	Value::CZString::isStaticString() const
1255	Value::isNull() const
1262	Value::isBool() const
1269	Value::isInt() const
1276	Value::isUInt() const
1292	Value::isDouble() const
1306	Value::isString() const

Таблица 2: Примеры методов с признаком «проверяющий метод» (файл json_value.cpp)

Стр.	Исходный текст
1261	bool
1262	Value::isBool() const
1263	{
1264	return type_ == booleanValue;
1265	}

Таблица 3: Исходный код метода признака «проверяющий метод»

При поиске шаблона «сравнивающий метод» (comparator) в тестовом проекте pugixml находятся методы, выполняющие функцию перегруженных операторов сравнения (Таблица 4).

Например, метод «operator==» в строке 3738 файла pugixml.cpp сравнивает поле «_attr» передаваемого по ссылке параметра «r» объектного признака «xml_attribute» и поле «_attr» данного объекта (Таблица 5).

Выводы

На основе УПП [2] и дополненного эквивалентного представления UIRDCF [7] была введена тестовая система признаков и разработан прототип утилиты, которая выделяет признаки квантов исходного текста согласно применению набора шаблонов (глоссария) над эквивалентным представлением UDCFM, группирует кванты, строит разбиения по заданным параметрам, тем самым выделяя нетривиальные

Стр.	Имя метода
3738	bool xml_attribute::operator==(const xml_attribute& r) const
3743	bool xml_attribute::operator!=(const xml_attribute& r) const
3748	bool xml_attribute::operator<(const xml_attribute& r) const
3753	bool xml_attribute::operator>(const xml_attribute& r) const
3758	bool xml_attribute::operator<=(const xml_attribute& r) const
3763	bool xml_attribute::operator>=(const xml_attribute& r) const
3974	bool xml_node::operator==(const xml_node& r) const
3979	bool xml_node::operator!=(const xml_node& r) const
3984	bool xml_node::operator<(const xml_node& r) const
3989	bool xml_node::operator>(const xml_node& r) const
3994	bool xml_node::operator<=(const xml_node& r) const
3999	bool xml_node::operator>=(const xml_node& r) const

Таблица 4: Примеры методов с признаком «сравнивающий метод» (файл pugixml.cpp)

Стр.	Исходный текст
3738	bool xml_attribute::operator==(const xml_attribute& r) const
3739	{
3740	return (_attr == r._attr);
3741	}

Таблица 5: Исходный код метода с признаком «сравнивающий метод»

множества признаков, а также позволяет выполнять поиск квантов по заданным признакам.

Предложенный подход выделения признаков показал свою применимость для задач поиска квантов исходного текста по сложным или плохо формулируемым условиям. Целью такого поиска может быть поиск недокументированных возможностей и потенциально ошибочных участков кода, при введении соответствующих признаков и категорий для искомых частей исходного текста (классов, методов, блоков).

Литература

[1] *Зубов М. В.* Статический анализ ПО с помощью его промежуточных представлений и технологий с открытым исходным кодом/М. В. Зубов, А. Н. Пустыгин, Е. В. Старцев//Матер. 2-й Междунар. конф. «FOSS. Lviv-2012», Львов. — Львів: Сорока, 2012. — С. 165–168.

[2] *Ошнуров Н. А.* Построение универсального промежуточного представления исходных текстов на языках C++ и C# / Н. А. Ошнуров, А. Н. Пустыгин, А. А. Ковалевский // Доклады Томского государственного университета систем управления и радиоэлектроники. — 2014. — Т. 33, № 3. — С. 135–139.

[3] Pugixml v1.2 — Light-weight, simple and fast XML parser for C++ with XPath support [Электронный ресурс]. URL: `http://pugixml.org/2012/05/01/pugixml-1.2-release.html` (дата обращения: 12.01.2017).

[4] Jsoncpp v0.5.0 C++ JSON parser [Электронный ресурс]. URL: `http://sourceforge.net/projects/jsoncpp/files/jsoncpp/0.5.0/` (дата обращения: 12.01.2017).

[5] LZ4 r131 — Extremely fast compression [Электронный ресурс]. URL: `https://github.com/Cyan4973/lz4` (дата обращения: 12.01.2017).

[6] Zlib 1.2.8 — A Massively Spiffy Yet Delicately Unobtrusive Compression Library [Электронный ресурс]. URL: `http://www.zlib.net/` (дата обращения: 12.01.2017).

[7] *Ковалевский А. А.* Построение эквивалентного представления исходных текстов программ в форме, пригодной для выполнения анализов потока данных в потоке управления / Ковалевский А. А., Пустыгин А. Н., Ошнуров Н. А. // Известия Юго-Западного государственного университета Серия управление, вычислительная техника, информатика. медицинское приборостроение. — 2015, №1 (14). — С. 28–34.

[8] *Пустыгин А. Н.* Построение эквивалентного представления зависимостей классов, полей, методов, функций и их перекрестного использования в исходных текстах программ./ А. Н. Пустыгин, А. А. Ковалевский, И. С. Белоусов //Одиннадцатая конференция «Свободное программное обеспечение в высшей школе» : Материалы конференции/Переславль-Залесский, 30–31 января 2016 года — 120 с.: илл. Тезисы доклада С. 40–44.

М. В. Зубов, А. Н. Пустыгин
Челябинск, Челябинский государственный университет

Прототип программного инструмента для анализа связности потока управления программ с открытым исходным текстом

Аннотация

Описано использования эквивалентных представлений исходного текста для получения участков связности потока управления на этапе статического анализа, для чего был разработан формат эквивалентного представления потока управления, пригодный для текстов на нескольких языках программирования. На основе этого формата написаны прототипы утилит получения такого представления, и на их основе разработаны прототипы утилит анализа, выходные данные визуализируются.

Анализ пути исполнения на этапе статического анализа — часто встречающееся действие при разработке, сдаче в эксплуатацию и сопровождении программ [4,5]. Использование универсальных промежуточных представлений позволяет применять унифицированный функционал для текстов на нескольких языках[1,2].

Для анализа потока управления исходного кода необходимо разработать соответствующее эквивалентное представление [3]. Поток управления — множество всех возможных путей исполнения программы[6].

Универсальное представление графа потока управления (*Universal Control Flow Representation, UCFR*) будет описывать только конкретный участок выполнения программы — функцию или метод класса. Назовем такой участок потока управления функциональным блоком. Граф потока управления является ориентированным графом общего вида, допускающим циклы.

Основным элементом потока управления является функция (метод класса), а для построения формата достаточно удержать в промежуточном представлении элементы (узлы) дерева разбора, приведенный в таблице для избранного круга языков программирования.

В качестве текстовой нотации эквивалентного представления выбран формат XML [7], что позволит сохранять эквивалентное представление в текстовом файле и проводить его анализ (возможно предварительный) вручную.

Язык	project	method/ function	block/ flow	tryExcept	finally	for	if	while	call
Java	+	методы		try/ catch	+	for/ foreach	+	while/ dowhile	+
Python	+	методы/ функции	+	try/ except	+	foreach	+	while	+
C++	+	методы/ функции	+	try/ catch	-	for	+	while/ dowhile	+
PHP	+	методы/ функции	+	try/ catch	(с версии 5.5)	for/ foreach	+	while/ dowhile	+

Таблица 1: Узлы эквивалентного представления избранных языков

Анализ графа вызовов целиком — задача, требующая значительных ресурсов и порождающая слишком много данных. Предложено ограничить анализ графа вызовов только отдельными участками или *трассами*. Трасса — это последовательность вызываемых функций или методов на графа потока управления. Определенная трасса соответствует некоторому возможному сценарию исполнения анализируемой программы и представляется в виде ориентированного графа. Вначале выделяются функциональные блоки, входящие в трассу. Начальной вершиной ребра, связывающего блоки, выступает вызывающий функциональный блок, конечной — вызываемый. Внутри функционального блока трасса также представляется в виде ориентированного графа, узлами выступают блоки, представленные в блок-схеме (БСА) данной функции/метода, располагающиеся вдоль трассы, ребрами — ребра БСА.

На основе функционала построения трасс были построены прототипы анализаторов для поиска создаваемых вдоль трассы объектов и для поиска вдоль трассы интересующих вызовов.

Выводы

Для выполнения статического анализа возможных путей исполнения программ по исходному коду был предложен формат универсального эквивалентного представления потока управления, на основе которого реализованы ряд прототипов анализаторов, предназначенных для графовых визуализаций. На основе этих моделей были предложены различные виды анализа трасс. Используя полученные графические модели можно анализировать исполнение программ с це-

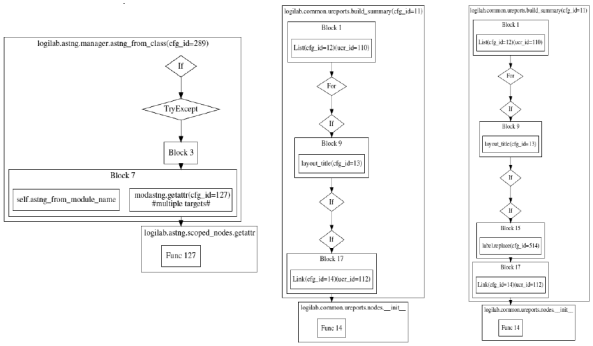


Рис. 1: Пример анализа трасс: визуализация трассы, поиск объектов, создаваемых вдоль трассы (объекты классов Link и List), поиск интересующих вызовов (функция replace).

лью обнаружения логических ошибок на ранних этапах разработки, а также исследовать функционирование и алгоритмы существующих программ.

Литература

[1] *Зубов М. В.* Построение универсального представления графа потока управления для статического анализа исходного кода /М. В. Зубов, А. Н. Пустыгин, Е. В. Старцев //Девятая конференция «Свободное программное обеспечение в высшей школе»: тез. докл. М.: Альт Линукс, 2014. с. 46–51.

[2] *Ошнуров Н. А.* Построение универсального промежуточного представления исходных текстов на языках C++ и C# /Ошнуров Н. А., Пустыгин А. Н., Ковалевский А. А. //Доклады Томского государственного университета систем управления и радиоэлектроники. — 2014. — Т. 33. — № 3. с. 135–139.

[3] *Пустыгин А. Н.* Построение эквивалентного представления зависимостей классов, полей, методов, функций и их перекрестного использования в исходных текстах программ /А. Н. Пустыгин, А. А. Ковалевский, И. С. Белоусов //Одиннадцатая конференция «Свободное программное обеспечение в высшей школе»: Материалы конференции/ Переславль — Залесский, 30–31 января 2016 года — 120 с.: илл. Тезисы доклада с. 40–44.

[4] *Зубов М. В.* Статический анализ ПО с помощью его промежуточных представлений и технологий с открытым исходным кодом /М. В. Зубов, А. Н. Пустыгин, Е. В. Старцев//Матер. 2-й Междунар. конф. «FOSS. Lviv-2012», Львов. — Львів: Сорока, 2012. — с. 165–168.

[5] *Зубов М. В.* Подходы к статическому анализу открытого исходного кода /М. В. Зубов, Е. В. Старцев, А. Н. Пустыгин //Сб. материалов Восьмой Междунар. конф. разработчиков и пользователей свободного программного обеспечения Linux Vacation/Eastern Europe. — Брест: Альтернатива, 2012. — с. 36–40.

[6] *Ковалевский А. А., Пустыгин А. Н., Ошнуров Н. А.* Построение эквивалентного представления исходных текстов программ в форме, пригодной для выполнения анализов потока данных в потоке управления //Известия Юго-Западного гос. ун-та, Серия управление, вычислительная техника, информатика. Медицинское приборостроение. 2015. № 1 (14). с. 28–34.

[7] *Зубов М. В.* Математическое моделирование универсальных многоуровневых промежуточных представлений для статического анализа исходного кода/М. В. Зубов, А. Н. Пустыгин, Е. В. Старцев //Докл. Томск. гос. ун-та систем управления и радиоэлектроники. 2014. Т. 33, № 3. с. 94–99.

Илья Захаров
Москва, SoftAccord

На каком уровне иерархии спецификаций начинается СПО?

Аннотация

Базовым понятием идеологии свободного программного обеспечения (СПО) является исходный код, полный доступ к которому — обязательное условие, без выполнения которого не возможно свободно использовать, модифицировать и распространять ПО.

Таким образом, исходный код ПО — это тот уровень иерархии спецификаций, отношение к которому явно и однозначно определено идеологией СПО. Подразумевается, что СПО соответствует также и открытым спецификациям (стандартам). Однако их иерархия и, соответственно, полный перечень не определен. Каждый проект СПО использует необходимый для него набор спецификаций.

Между тем, на разных уровнях различных иерархий спецификаций происходят интенсивные процессы: спецификации разрабатываются, развиваются и предлагаются к использованию, в том числе на проприетарных условиях. Некоторые спецификации значительно влияют на развитие информационных технологий в целом и СПО в частности. Например, концепция обобщенного программирования и MDA. Для формирования обоснованных выводов о соотношении идеологии СПО и процессов разработки спецификаций необходимо определить иерархию, в рамках которой это соотношение возможно оценить. В докладе предлагается рассмотреть подходы к решению проблемы оценки различных спецификаций с точки зрения идеологии СПО.

1. Возможное определение иерархии спецификаций для оценки их соотношения с идеологией СПО

Построение такой иерархии должно учитывать основные направления развития информационных технологий (ИТ) и включать уже сложившиеся фрагменты структуры перспективных спецификаций. Обосновать легче наиболее общий уровень, начало иерархии, т. к. его можно определить из опыта, на основании исследования направлений развития ИТ. Он же, как родительский класс определит нижележащие слои иерархии спецификаций.

Например, вот что сказал Бьерн Страуструп в интервью Cnews[1]в 2010 году: «Когда я начинал, никто не слышал об объектно-ориентированном программировании, а почти все, кто «знал», о чем шла речь, считали тему невозможной (слишком сложно, слишком трудно изучить, слишком медленно, слишком специализировано и так далее), и все же сегодня мы все пользуемся результатами той идеи. Аналогичная ситуация наблюдается сейчас с парадигмой обобщенного программирования: только представьте, что будет через десять лет...».

Здесь, как и во многих других авторитетных источниках речь идет об объектно-ориентированном программировании как стратегическом выборе. На сегодняшний день этот выбор привел к тому, что сформировалась иерархия спецификаций, которые в сбалансированном виде входят в язык моделирования UML[3]. Причем, эту иерархию спецификаций есть все основания выстраивать начиная с онтологического понятия «объект», в смысле объект познавательной деятельности[2], то есть рассматривая объектно-ориентированный анализ как средство научной познавательной деятельности. Эти вопросы кратко изложе-

[1] `http://www.cnews.ru/articles/eksklyuzivnoe_intervyu_s_sozdatelem`

ны в докладе[1]. В такой трактовке верхние уровни иерархии спецификаций ИТ-проектов могут выглядеть так, как изображено на рисунке 1.

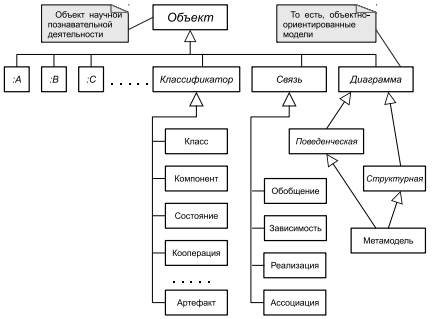


Рис. 1: Объект научной познавательной деятельности и объектно-ориентированный анализ

2. Примеры кратких спецификаций диаграмм (моделей)

Дальнейшее специфицирование под свободными лицензиями типов моделей, называемых в языке UML диаграммами, создает методологические и юридические основания для развития открытой коллективной разработки ПО.

2.1 Краткая спецификация (теоретическая схема[2]) диаграммы последовательности (вида диаграммы взаимодействия), которая в настоящее время подразумевается в языке UML[3].

2.1.1 Диаграмма взаимодействия — модель поведения кооперации, реализующая интегративное свойство системы. Другие варианты контекста: взаимодействие между объектами в реализации операций и ВСКД семантики класса.

2.1.2 Способы моделирования взаимодействий:

в контексте времени (диаграмма последовательности);

в контексте структурных связей (диаграмма коммуникации).

2.1.3 Взаимодействие (interaction) — поведение, в которое вовлечено множество сообщений, передаваемых между объектами, которые играют определенные роли в некотором контексте для достижения заданной цели.

2.1.4 Сообщение — это спецификация взаимодействия объектов, которая передает информацию и ожидает последующих действий.

Сообщение чаще всего вызывает операции или передачу сигнала, а также может сопровождаться созданием или уничтожением объектов.

2.1.5 Типы сообщений :

call (вызвать) — вызов операции объекта (наиболее частое событие);

return (возвращает) — вызывает сообщение вызывающему объекту;

send (послать) — посылает сигнал объекту;

create (создать) — создать объект;

destroy (уничтожить) — уничтожает объект, в т.ч. сам себя;

не специфицируемые в UML сообщения (в части синтаксиса и семантики).

2.1.6 Синхронное сообщение — сообщение, ожидающее ответа (стрелка с закрашенным треугольником), ответ на синхронное сообщение (возврат из вызова) — пунктирная стрелка уголком (может не отображаться).

2.1.7 Асинхронное сообщение — сообщение, не ожидающее ответа (стрелка с уголком).

2.1.8 Роль — прототип объекта.

2.1.9 Коннектор — прототип связи между ролями.

2.1.10 Ссылка (link) — экземпляр ассоциации, смысловое (семантическое) соединение объектов друг с другом.

2.2 Краткая спецификация (теоретическая схема[2]) диаграммы состояния (или конечного автомата), которая в настоящее время подразумевается в языке UML[3].

2.2.1 Диаграмма состояний (state diagram) — модель конечного автомата, показывающая поток управления от одного состояния к другому (граф с вершинами и ребрами). Может использоваться для моделирования практически любого элемента системы, однако чаще всего применяется для моделирования всей системы, подсистемы, класса, сценариев варианта использования, поведения интерфейсов.

2.2.2 Конечный автомат (state machine) — описание последовательности состояний, через которые проходит объект на протяжении жизненного цикла, реагируя на события, а также описание реакции на эти события. Подходит для спецификации поведения объекта, который должен отвечать на асинхронные сообщения либо его поведение зависит от прошлого.

Явно специфицируются: события, на которые объект может реагировать, реакция на эти события, влияние более раннего поведения на текущее.

2.2.3 Состояние (state) — ситуация в жизненном цикле объекта, на протяжении которой он удовлетворяет некоторому условию, выполняет некоторую деятельность или ожидает некоторого события.

2.2.4 Событие (event) — спецификация существенного факта, которое происходит во времени и пространстве. В контексте автомата событие — это воздействие, которое вызывает переход между состояниями.

2.2.5 Переход (transition) — связь между двумя состояниями, показывающая, что объект, находящийся в первом состоянии, должен выполнить некоторые действия и перейти во второе, как только произойдет определенное событие и будут выполнены определенные условия.

2.2.6 Деятельность (activity) — процесс, работа, происходящая внутри автомата.

2.2.7 Действие (action) — примитивное выполняемое вычисление, приводящее к смене состояния модели или возврату значения.

2.2.8 Характеристики состояния объекта (автомата объекта):

- имя — текстовая строка, отличающая данное состояние от других (состояние может быть анонимным, т.е. без имени);
- входной/выходной эффекты (entry/ exit) – действия, выполняемые при входе/выходе из состояния;
- внутренние переходы (internal transition) — переходы, которые обрабатываются без смены состояния;
- подсостояния (substate) – вложенные состояния, в том числе неортогональные (последовательно активные) или ортогональные (параллельно активные);
- отложенные события (deffered event) – список событий, которые не обрабатываются в данном состоянии, но откладываются и помещаются в очередь для обработки объектом в другом состоянии.

3. Спецификации метамоделей, начиная с наиболее общего (абстрактного) уровня

3.1 На рисунке 2 представлена предельно общая структурная модель современной научной картины мира. Её особое значение, несмот-

ря на кажущуюся тривиальность, заключается в том, что данное представление является фактическим консенсусом.

Рисунок 2. Структурная модель объекта «Современная научная картина мира».

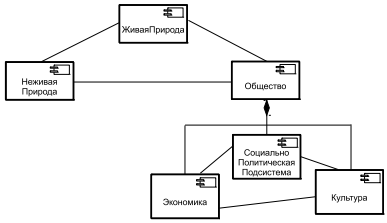


Рис. 2: Структурная модель объекта «Современная научная картина мира».

Степень общности и полноты, таким образом, определяет её прикладное значения для разработки различных систем. Если для каких-то задач целесообразно изобразить эту схему как на рисунке 2 в виде компонентов (концептуально более мягкая форма — пакеты), то в смысле объектно-ориентированного проектирования надо согласиться, что эта схема изображает структуру родительского класса для любых других систем, которые так или иначе должны наследовать свойства и операции этого класса. Вне терминологии объектно-ориентированного анализа и программирования то же самое можно сказать так: эта структура в явном или неявном виде должна учитываться при разработке любых теорий (схем, моделей и т.п.).

3.2 Вероятно, наиболее перспективная метамодель, позволяющая составить формализованное представление о сложной системе изображена на рисунке 3.

Эта метамодель отражает подход к моделированию сложной системы, когда в качестве первой спецификации удалось выявить отдельные компоненты системы и определить их взаимосвязи. Стохастические процессы и другие неформализуемые характеристики остаются внутри компонентов, которые как «черные ящики» могут в дальнейшем быть более подробно специфицированы через состояние портов.

3.3. Распространенной метамоделью является первичное представление о сложной системе, как о логически разделяемом представлении

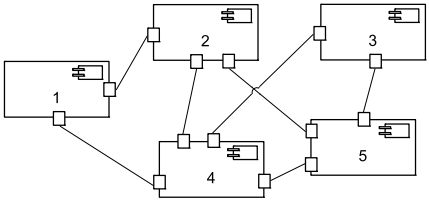


Рис. 3: Условное название метамодели «внутренняя сложность».

о ядре и его окружении (или периферии). На рисунке 4 эта метамодель изображена в виде совокупности «пакетов» между которыми существуют связи типа «зависимость». «Пакеты» являются логическим представлением проектировщика о системе. На физическом уровне в системе такое разделение может отсутствовать.

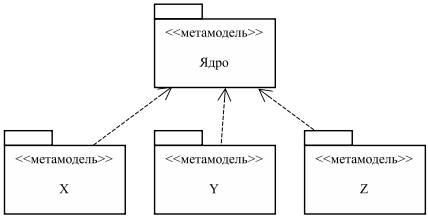


Рис. 4: Метамодель «ядро — периферия».

3.4 На рисунке 5 приведена поведенческая метамодель (в авторской нотации), которая отражает сценарий поведения систем уровня системной сложности с самоорганизацией (саморазвитием), предложенная В.С. Стёпиным[2].

1. Исходная саморегуляция.

2. Новый тип саморегуляции, основанный на трансформации предшествующих уровней иерархии системы.

3. Потенциально возможный уровень организации при продолжении развития системы как возможность нового типа саморегуляции.

Выводы

1. В терминах иерархии спецификаций, изображенной на рисунке 1, исторически сложившаясяся, «естественная», конкуренция свободного и проприетарного ПО находится в области спецификаций конкретных Классов, Объектов (в смысле экземпляров

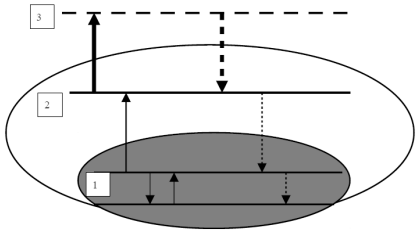


Рис. 5: Общая поведенческая модель самоорганизующихся систем, предложенная В.С. Степиным

классов) и Артефактов. Расширение этой области конкуренции может привести к негативным для развития ИТ последствиям.

2. Целесообразно вовлечение в сферу свободных лицензий максимального объема спецификаций, начиная с максимально обобщенных уровней. Предположительно, наиболее важным на начальном этапе является публикация метамоделей различного рода и уровня иерархии, включая профили, шаблоны и т.д.

3. Требуется дальнейшая проработка вопросов построения иерархии(-ий) спецификаций, в рамках которых возможно определение отношения различных документов (спецификаций, стандартов) с идеологией СПО.

4. Если строить иерархию спецификаций от уровня Объекта научной познавательной деятельности, то это может создать юридические основания для защиты открытости нижележащих спецификаций, так как они наследуют его свойства и операции.

Литература

[1] *Захаров И. В.*, XI конференция «Свободное программное обеспечение в высшей школе»: Материалы конференции; Метаданные языков визуализации, специфицирования, конструирования и документирования (языки ВСКД) на примере UML/SysML. — М.: Альт Линукс, 2016. — 120 с., ISBN 978-5-905167-20-1

[2] *Степин В. С.*, Теоретическое знание. — М.: Прогресс-Традиция, 2000. — 744с., ISBN 5-89826-053-6

[3] *Grady Booch, James Rumbaugh, Ivar Jacobson*, Введение в UML от создателей языка. — М.: ДМК Пресс, 2012. — 494 с., ISBN 978-5-94074-644-7

Стас Фомин
Москва, Virtuozzo https://virtuozzo.com/, ИСП РАН http://ispras.ru, http://discopal.ispras.ru МФТИ http://mipt.com

Эффективная «домашка» — задачи студентам на MediaWiki

Аннотация

В докладе представлен опыт эффективной организации практической проверки знаний в виде задач, в курсах «Эффективные алгоритмы» и «Сложность алгоритмов», для студентов ФУПМ МФТИ и студентов с базовой кафедрой Института Системного Программирования РАН. Почти десять лет обкатывался опыт «коллаборативного» решения задач на портале курсов http://discopal.ispras.ru, реализованного на MediaWiki4IntraNet — расширении общеизвестной MediaWiki, с реализованной системой прав и множеством расширений для удобного ведения научно-учебного контента. О том, почему это эффективно и о деталях реализации, мы и поговорим.

Дистанционный ли курс или оффлайн, читаются ли лекции и семинары лично, или отсылают к своим или чужим записям или книгам — всегда стоит вопрос эффективной тренировки, организации домашней работы студентов. Да, тут существуют и простые решения — например «тесты», которые, к сожалению, и считаются «практической проверкой» в большинстве дистанционных курсов. Но если we need to go deeper, то, если не рассматривать специализированные системы лабораторных работ, которых трудно сделать универсальными, самой привычной системой, подходящей для любых естественнонаучных курсов, являются классические задачи, требующие оформленного решения — текст, формулы, картинки-псевдокод.

Вопрос в том, как организовать эти «задачи-решения», чтобы была «Win-Win» ситуация и для студентов, и для преподавателей.

Студентам, чтобы...

- было интересно, чтобы было видно, что они не просто делают что-то, попадающее сразу в мусорную корзину, но что-то

нетленное-полезное, а ситуация побуждала их самостоятельно приступать к участию, с максимальной свободой выбора;

- было удобно, нестыдно и быстро оформить решение, получив отличное форматирование не только для текста, но и для формул, кода, графики.
  - при этом, избежать изучения чего-то совсем бесполезного в будущем — например, интерфейса каких-то самодельных систем.
  - получение опыта, полезного потом в индустрии — это скорее, плюс.
- все было прозрачно и наблюдаемо — а не «мои решения потерялись, я высылал», «почему не засчитали, где я был неправ?», «это не я списал, а у меня».

Преподавателям:

- Отсутствие читинга — списываний и т.п.
- Минимальная нагрузка при проверке — чтобы не было возьни с тетрадками, теряющимися сообщениями в почте, переписки,
  - плюс, те же требования к удобству при оформлению решений — если в нужно в решении показать, где студент неправ.
- Легкость оформления задач.
- Возможность делегировать проверку (P2P).
- Откуда-то брать эти задачи.
- Геймификация — для студентов старших курсов, особенно по выбору, нужна мотивация, естественным образом побуждающая их включится в изучение курса заранее (а не за два дня до экзамена).
- Ну и чтобы все можно было это сделать просто и понятно, не заводя каких-то специальных узкоспециализированных непонятных систем.

Мы реализовали все это на базе MediaWiki4IntraNet — расширении общеизвестной MediaWiki, с реализованной системой прав и множеством расширений для удобного ведения научно-учебного контента. Детали реализации мы и покажем в докладе.

Георгий Курячий
Москва, ВМК МГУ, ООО "Базальт СПО"

И ропщет мыслящий тростник

Аннотация

Второй год преподавания дисциплины «Алгоритмы и алгоритмические языки» в Севастопольском филиале ВМК МГУ довольно определённо показал достоинства и недостатки использования языка программирования Python3 в рамках данного курса и данной аудитории. В докладе обсуждается опыт преподавания: подсказанные Python3 и найденные самостоятельно методические приёмы а также существенное увеличение объёма необходимого учебного материала по сравнению с аналогичным курсом на базе языка программирования Pascal.

В режиме тезисов

1. Состояние дел на факультете в 2015: Паскалисты против Сишников
   - 80-е: Algol → Паскаль
   - Разработка Л.С. Корухова, В.Н. Пильщиков (каф. АЯ)[1]
     - 4 лекции — понятие алгоритма, МТ, НАМ, проблема останова
     - 11 лекций — Паскаль
     - 6 лекций — структуры данных (списки, стек/очередь, деревья, таблицы ссылок и хеш-таблицы)
     - Практикум
   - Всё начало 2000-х: «давайте заменим Паскаль на что-нибудь»...
     - (каф СП) «...на Си!»
     - см. выступление В. П. Иванникова на нашей конференции в 2013-м году[2]

[1] `http://uneex.ru/PascalAAL`

[2] `http://0x1.tv/Использование_СПО_в_образовании_(Виктор_Иванников,_OSEDUCONF-2013)`

2. Доклад Python Domination[3] на той же конференции (слайды[4]).
   - Универсальность
   - Актуальность
   - Простота
     - В т. ч. в качестве первого ЯП
3. 2014-2015 у. г.: спецкурс «Язык программирования Python[5] / Разработка прикладных программ на языке программирования Python[6]»
   - Популярность
   - Хороший выход годного
   - Опыт использования EJudge
   - Опыт сопровождения скринкастами
   - Вывод: хороший, годный язык!
4. 2015 у. г. I семестр и 2016 у. г. I семестр: базовый курс «Алгоритмы и Алоритмические языки» в Севастопольском филиале ВМК МГУ[7]
   - Дистанционные лекции
     - Использование эмуляторов везде, где возможно (НАМ, МТ, СД, в планах — РФ и БНФ)
   - Дистанционные «семинары» (не практикум! 2016 г. — практикум отдельно)
     - С использованием Moodle chat
   - Использование Moodle и EJudge
     - Домашние задания (наполовину посредством EJudge, наполовину — «эссе» в Moodle)
     - Учёт ДЗ и контрольных при выставлении оценки за экзамен
   - Связь с преподавателем — группа ВКонтакте

[3] `http://0x1.tv/Использование_языка_программирования_Python_в_качестве_базового_при_обучении_специалистов`

[4] `http://uneex.ru/slides.com/frbrgeorge/python-domination`

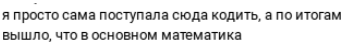
[5] `http://uneex.ru/LecturesCMC/PythonIntro2014`

[6] `http://uneex.ru/LecturesCMC/PythonDevelopment2015`

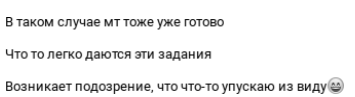
[7] `http://uneex.ru/FrBrGeorge/RemoteClasses2016`

- Итоговый «семестровый проект»: написание на Python эмуляторов МТ, НАМ и БНФ-парсера
- Дистанционный экзамен (много «автоматов»)

5. Итоги
   - низкий уровень абирутриентов
     - Что такое БНФ — оба раза не поняли (лектор виноват?)
   - Отсутствие мотивации и некорректный таргетинг (особенно 2016 г.)
     - На семинарах работают 2-3 человека max
     - Самая умненькая девочка отчислилась:
   - Копипаста

   - Объём необходимых знаний: Python >> Pascal
   - До темы «деревья» оба раза просто не дошли
   - «Семестровый проект»:
     - Чуть ли не все сделали эмуляторы МТ и НАМ
     - Мнение студента, который просто вовремя делал все задания:

     - 20% сделали БНФ-парсер
6. Что делать?
   - Объём против уровня подготовки
     - Усекать Python (изобретать Питончик)?
     - Разделять Python для всех и доп. главы?
     - Учить именно программированию, как в техникуме?
     - Другой ЯП? А какой?
   - Мотивация
     - Геймификация (чтоа?)
       - Turtle или что-то такое
       - Соревнования и рейтинги
     - Реорганизация семинара:

  - Официализация
  - Тривиальные упражнения вместо задач
  - Отчёт всех участников
  - Оценки за семинар
- Контроль: пример усердного студента

М. В. Быков
Москва, http://diglossa.org

Проект: Морфологический анализатор Морфей http://gr.diglossa.org

Морфологический анализатор древнегреческого на основе electron.js

Аннотация

Приложение Морфей для древнегреческого языка работает автономно, без подключения к сети, на трех платформах, Linux, MacOS, Windows, и в любом месте на десктопе, — достаточно скопировать греческое предложение в буфер обмена. Используются несколько специальных и несколько общедоступных словарей. Анализируется все предложение, включая простые синтаксические связи между словами.

πολυμαθίη νόον ἔχειν οὐ διδάσχει
многознание уму не научает
Гераклит

Большие проекты изучения национальных языков находятся на очевидном свободном взлете — только дайте денег, и они станут еще больше и мощнее, пределов не видно. Для древнегреческого самый большой — Персей [1]. Возьмем, к примеру, слово — *λόγος*. Исходное его значение — складывать, собирать вместе. Его слышно в наших словах ложка, телега. Что говорит о слове *λέγω* — я говорю — Персей [2]? Откроем страницу. Мы видим большое количество вариантов. Как этим пользоваться? Первое значение с 8 морфологическими расшифровками — *λέγαι γυναῖκες* — lewd — развратные девки. Ясно, что автор выражения (Архилох) имеет в виду болтушек (*λόγος*) — кумушек. У слова болтушки есть еще значения сплетницы и сводницы. Сплетать-сводить — то то же самое складывание, что и в исходном слове *λόγος*. В любом слове всегда слышится и противоположное значение, в данном случае не сплетение, а раз-врат, то есть рас-

кладывание. T.e. lewd — действительно, возможное значение. Бибихин об присутствии в слове противоположных смыслов пишет как о "сне языка". Ясно, что на Персее мы видим одно из возможных поэтических-образных истолкований слова $\lambda\acute{o}\gamma o\varsigma$, которому придана грамматическая форма и выделена ячейка в базе данных словаря. И выполнен отдельный морфологический анализ. Очень перспективный метод для тех, кто ищет работу в этой области. Через пару лет можно ожидать появления еще нескольких разделов-результатов для слова $\lambda\acute{\varepsilon}\gamma\omega$. Будущим студентам-линвистам будет что преподавать. Создателей ресурса можно поздравить.

Однако это все довольно подозрительно. Авторы словаря Liddell, Scott и Jones и создатели Персея считаются, вроде бы, английскими джентельменами, но действуют шашкой подобно кавалеристам Семена Буденного.

Но дело еще хуже. В древнем языке, ни в Греции, ни в Индии, вообще нет ни существительных, ни прилагательных. Согласно Дионисию Фракийцу, имя имеет 19 видов. Среди них нет ни существительных, ни прилагательных. И нельзя объединить несколько видов вместе, чтобы получить наше привычное прилагательное — придется либо исключить некоторые прилагательные, либо включить существительные. Я об этом здесь говорить не буду, кто хочет посмотрите сами [3]. Дело в ином — в античности, и в Индии также, речь идет вообще не о языке — такого понятия нет, это позднее, средневековое, если не после-декартовское понятие. А о речи. А речь — это не язык, речь, когда шевелятся губы, а язык — это теоретический конструкт. И принципы деления речи на "части" и выделения "частей речи" в нашем языкознании — разные и логически несовместимые.

Другими словами, все современные корпуса текстов, во главе с Персеем, ни к черту не годятся, если вы хотите понять, о чем говорит древний автор. Они занимаются созданием современной грамматической теории древнего языка. Наверное, это кому-то интересно, раз им деньги дают. Предлогаю назвать вышеописанный механизм "злокачественным распуханием грамматической теории", ЗРГТ.

Это я говорю, чтобы была понятна цель разработки Морфея. Морфей создан для автоматизации процесса понимания текста, а не для изучения языка. В процессе понимания очень много услилий приходится тратить на листание и чтение огромных словарей, рытье в толстенных грамматиках и чесание в затылке. Все это вполне автоматизируемые процессы.

Посмотрите скринкаст на странице http://gr.diglossa.org

Морфей работает в любой большой ОС (не работает пока на мобильных платформах), и в любом месте на десктопе — достаточно выделить и скопировать текст в буфер обмена (обычно ctrl-c). Это приложение electron.js [4], то есть по сути браузер Хромиум. Морфей может работать автономно, он имеет встроенную базу данных Pouch.js, но при каждом запуске, если есть выход в сеть, локальная и серверная база синхронизируется прозрачно для пользователя.

Морфей имеет модульную структуру. Основной процесс такой:

- все слова в тексте (обычно предложение, или клауза — от знака препинания до знака препинания) проверяются в словаре "терминов". Термины — это конечные формы, не требующие дальнейшего анализа. Например, все формы местоимений, артиклей, все неизменяемые формы в словаре, неправильные формы глаголов и т.д. Поскольку греки записывали речь, а не "информативный текст", они записывали все, что слышали. Например, часть фразы *εἲ περ γάρ ἐστιν ἠψυχή ἐν* имеет только одно слово не-термин — *ψυχή*, а все остальные — частицы, предлоги, артикли и форма глагола "есть". Это очень типичный случай.

Для оставшихся изменяемых слов по окончаниям вычисляются вероятные словарные формы. Для этого форма пропускается через каскад модулей-фильтров. Количество фильтров будет наращиваться в следующих версиях. Наличие вероятных словарных форм проверяется в словаре.

В качестве большого словаря я использую свободно доступный в сети словарб YALS — "Yet Another Liddell-Scott" с неясной лицензией. Словарь доступен здесь [5]. Видимо, он имеет какое-то отношение к словарю Лидделл-Скотта. Для найденных в словаре вариантов по окончаниям вычисляются морфологические характеристики.

Дополнительные технические словари терминов и конечных форм я создал вручную и эта работа будет продолжена, источники [6]. В отличие от "больших" морфологических анализаторов я не пытаюсь выделить все возможные лексические единицы слова, но хотел бы вывести слово из его источника. В идеале я хотел бы иметь несколько уровней словаря:

- (пра) — индоевропейский корень
- язык Гомера
- классическое значение
- койне

- ново-греческий

где каждый уровень представлен двумя-тремя типичными значениями, не в качестве установленного правила, но лишь для наводки на тему. А затем читателю должны быть представлены примеры переводов из живой переводческой практики. На различные известные языки в различных контекстах. Это просто выполняется поиском по параллельному массиву текстов. Сведение анализа к единому значению, вместо создания для каждого значения слова своей лексической единицы, как мне кажется, поможет справиться с ЗРГТ.

Далее между словами предложения устанавливаются типичные связи. Связанные слова подчеркиваются. Морфей не анализирует синтаксическую структуру предложения — это конструкт современной теории языка. Древний автор не знал о синтаксической структуре. А согласованные слова — вполне себе были. Пока что выделяются и подчеркиваются только связанные артикли, местоимения, имена.

Поскольку Морфей имеет модульную структуру, любые модули могут быть заменены. И на этой основе создано иное приложение. Например, в стиле более современной грамматической теории. Или с подключением иных словарей. Или с иной локализацией. Etc.

В целом Морфей, я надеюсь, может несколько облегчить процесс чтения греческого текста. Но Морфей по своей конструкции и цели не может выдать готовый перевод. Перевод, и понимание одного слова и всего текста — это поступок, уникальное событие в мире, и совершает его только читатель, и только в процессе чтения.

Литература

[1] http://www.perseus.tufts.edu

[2] http://www.perseus.tufts.edu/hopper/morph?l=λέγω&la=greek#lexicon

[3] https://el.wikisource.org/wiki/Τέχνη_Γραμματική

[4] http://electron.atom.io/

[5] http://diglossa.org:5984/_utils/database.html?yals

[6] https://lrc.la.utexas.edu/eieol_master_gloss/grkol/2

[7] http://www.chlt.org/FirstGreekBook/

[8] http://biblehub.com/englishmans_greek.htm

Карпеш С. В., Петров А. А.
Переславль-Залесский, ЯрГУ им. Демидова

Разработка и внедрение системы управления погружным жидкостным охлаждением высокопроизводительного кластера

Аннотация

Наряду с традиционными воздушными системами охлаждения все большее распространение получают жидкостные. Жидкостное охлаждение более эффективно благодаря большей теплоемкости, коэф. термопередачи и т.д. Ещё одним плюсом является то, что жидкостные системы охлаждения гораздо компактнее традиционных воздушных кулеров и имеют меньшее энергопотребление. Однако, функционирование погружной системы охлаждения невозможно без дополнительного потребления энергии: насосы, перекачивающие охлаждающую жидкость, вентиляторы драйкулера. В статье рассмотрен процесс разработки системы управления погружным охлаждением с использованием открытого программного обеспечения (в узлах системы управления и на этапе разработки узлов).

Современный высокопроизводительный кластер — сложная система, генерирующая огромное количество тепла. Если его не отводить, то последствия будут весьма серьезные. При этом выход из строя вычислительного модуля не является самым страшным, ведь при перегреве есть риск потерять данные, что можно смело назвать катастрофой. На рынке множество вариантов систем охлаждения, но все они дорогие. Причем термин дороговизна здесь применен не к стоимости установки, а к стоимости обслуживания. В случае с воздушными системами охлаждения, расходы на охлаждение составляют примерно 60% от расходов на сами расчеты (Power usage effectiveness 1, 6). [1] Гораздо лучше обстоят дела с погружным жидкостным охлаждением, где показатель PUE держится на отметке 1.04. [2] Однако такая система сложна в проектировании и управлении. Она содержит бак, куда в дальнейшем будет установлено вычислительное оборудование, теплообменник, в котором от хладагента тепло передается воде (более дешевой, но при этом более теплоемкой) и выносной драйкулер, где тепло передается в атмосферу. В случае, когда драйкулер может находиться на небольшом расстоянии от бака, возможен отказ от водяного контура.

Главным плюсом погружной системы охлаждения, с точки зрения разработчика, является её инертность. В случае возникновения какой либо неисправности в системе управления, у неё будет время на восстановление, так как большой объем достаточно теплоемкой жидкости быстро нагреть до критической отметки физически невозможно.

Нами была разработана система, состоящая из двух частей. Первая занимается только управлением исполнительных устройств и включает в себя промышленный контроллер, набор датчиков температуры, давления, а так же устройства (преобразователи частоты) для управления насосами и вентиляторами. Вторая часть системы занимается визуализацией и логгированием данных.

Сердцем аппаратного комплекса системы управления охлаждением является промышленный контроллер OWEN ПЛК-150. На борту имеет цифровые интерфейсы RS-232, RS-485, Ethernet. Весь обмен между ПЛК и внешними контроллерами выполняется по протоколу Modbus. Всего в наличии один преобразователь частоты на вентиляторы, два для управления насосами и контроллер с дополнительными аналоговыми портами для подключения датчиков давления.

Для корректной работы алгоритма управления необходимо знать температуру в баке. Но при этом датчики температуры должны располагаться рядом с процессорами, где жидкость будет более горячей. К сожалению использовать показания температуры с датчиков, встроенных в процессоры затруднительно, так как придется вмешиваться в работу кластера. Отсюда следует, что датчиков температуры должно быть достаточно много. При использовании наработок производителей устройств для промышленной автоматизации возникает серьезная проблема, связанная с монтажом. Все производители используют аналоговые датчики, что подразумевает подведение пары проводников к каждому такому термометру. Учитывая их количество и количество входов на ПЛК (придется использовать внешние контроллеры ввода-вывода) получаем очень громоздкую и сложную в монтаже конструкцию. В рамках этого проекта был разработан контроллер для опроса множества (теоретический максимум 2^28) цифровых термометров, что заметно упростило систему.

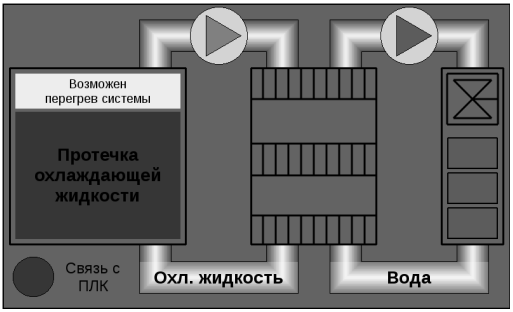
Алгоритм управления разделяет систему на три контура. Первый контур представлен максимальной температурой бака и насосом охлаждающей жидкости. Температурой бака служит максимум из массива всех температур в баке. Данный контур управляет температурой процессоров путем изменения скорости циркуляции хладагента. Второй

контур состоит из водяного насоса, датчиков давления и температуры. Контур позволяет управлять температурой хладагента путем изменения скорости циркуляции воды. Последний контур управляет температурой воды при помощи изменения потока воздуха. Он состоит из датчиков температуры на входе и выходе драйкулера и вентилятора. Использование преобразователей частоты позволяет более точно управлять исполнительными устройствами, что в итоге дает ещё большее снижение затрат.

Визуализация состояния кластера и сохранение данных выполнено в OpenSCADA. Она является открытой SCADA системой, построенной по принципам модульности, многоплатформенности и масштабируемости. [3] В качестве платформы выступает одноплатный компьютер Raspberry Pi 3 версии с одной небольшой модификацией (удален чип, отвечающий за беспроводные интерфейсы). Используется операционная система Debian. Две части системы соединяются при помощи одного провода Ethernet. У Raspberry имеется два сетевых интерфейса, одним из которых она соединена с ПЛК, а вторым с локальной сетью объекта, где установлен кластер. OpenSCADA запрашивает состояние системы из ПЛК по протоколу Modbus TCP. Все эти данные в неизменном виде сохраняются на внешний накопитель на случай, если придется анализировать работу системы при сбое. Сохраняются показания всех датчиков температуры, информация об уровне жидкости, работе насосов и вентиляторов. Часть этих данных выводится на экран. Так как состояние системы должно быть видно с большого расстояния, а физические размеры дисплея достаточно малы, то отображать много данных, и вообще какие либо числовые параметры, не имеет смысла. Учитывая эти особенности была разработана мнемосхема.

При необходимости средствами OpenSCADA реализован удаленный контроль по сети через Веб-интерфейс. Так как используется полноценный Linux, то можно легко обезопасить систему от проникновения из вне множеством различных способов.

В результате была разработана система управления погружным охлаждением высокопроизводительного кластера, достойно показавшая себя на испытаниях. На данный момент система подобного типа используется в нескольких проектах специального назначения, один из которых рассчитан на работу в экстремальных условиях (до $-50°C$).



Литература

[1] *Абрамов С. М., Амелькин С. А.* Энергоэффективность высокопроизводительных вычислительных комплексов: анализ современных достижений Российской суперкомпьютерной отрасли // НСКФ, Переславль-Залесский, 2016

[2] *Амелькин С. А., Чичковский А. А., Абрамов С. М., Клюев Л. В.* Вычислительные кластеры Иммерс — инновации и практика использования. НСКФ, Переславль-Залесский, 2015

[3] Официальный сайт OpenSCADA `http://oscada.org/`

Хаткевич М. М.
Переславль-Залесский, Исследовательский центра медицинской информатики ИПС РАН

Моделирование прямоточного сумматора

На сегодняшний день темпы развития вычислительных возможностей сократились вследствие наличия физических ограничений транзисторных процессоров. Идет поиск решения, которое позволит увеличить вычислительные мощности с минимальными затратами. Одной из проблем в вычислительных устройствах является стена памяти. Н. Н. Непейвода разработал прямоточный сумматор, где вычисления передаются между вычислительными узлами без хранения промежуточных результатов.

На основе схемы сумматора Н. Н. Непейводы была реализована программная модель, способная работать с разными типами данных.

Модель сумматора была запущена для работы с числами в различных системах счисления, а именно: двоичной, восьмеричной, симметричной троичной. Также были проведены испытания на двоичных матрицах и действительных числах.

Установлено, что схема прямоточного сумматора Н. Н. Непейводы инвариантна для всех перечисленных типов данных.

Планируется доработать модель для работы в интервальной системе счисления.

Алексей А. Цветков
Переславль-Залесский, ИСП РАН, ГК «Интерин»

Фрактальное моделирование систем кровеносных сосудов

Аннотация

Предлагается феноменологическая модель электромагнитного излучения человека на уровне организма и его взаимодействия с другими системами. Одним из базовых компонентов модели являются фрактальные функции, которые позволяют представить органы и системы органов организма в формализованном виде. Описывается методика формирования фрактальных моделей организма на основе L-систем на примере систем кровеносных сосудов. Предлагаемая методика может использоваться не только в биологии и медицине - диапазон применения может быть расширен практически на все предметные области, объекты которых содержат фракталы. На основе представленного материала может читаться курс по фрактальному моделированию для студентов старших курсов естественно-научных факультетов университетов.

Неоспоримым фактом является то, что организм состоит из атомов и молекул, которые содержат положительно и отрицательно заряженные частицы, и, которые могут перемещаться по телу (например, поток крови или лимфы) или совершать колебательные движения (например, в клетках, образующих органы). Как правило, элементы, формирующие организм, представляют собой, так называемые, диполи, которые в границах части органа, органа или системы органов

могут действовать, как ансамбль, т.е. образуется один макроскопический дипольный момент, который эквивалентен двум заряженным с противоположным знаком поверхностям на границе. При этом, внутри среды все заряды скомпенсированы. Из этого следует, что напряженность поля отдельных частиц или частей организма может суммироваться и достигать значений в мкВ и выше, что вполне достаточно для регистрации современной аппаратурой, как вблизи, так и на значительных расстояниях: от десятков метров до сотен километров. Все зависит от диапазона электромагнитного излучения (далее ЭМИ), в котором происходит излучение.

Другим фактом, связанным со строением организма, является его фрактальность, которую подметили многие специалисты, занимающиеся фракталами. Пример моделирования легких фракталами привел Б. Мандельброт в 1977 году. Но в данном случае автор использовал недостаточное количество итераций, что объяснимо тогдашними возможностями вычислительной техники. Т.е., на основе приведенных выше фактов можно построить феноменологическую фрактальную модель, которая позволит использовать радиотехнические методы для расчета диапазона и интенсивности ЭМИ для каждого из органов или систем органов, что позволит получить интегральную амплитудно-частотную характеристику (далее АЧХ) организма, которая, в свою очередь, может позволить получать диагностическую информацию о состоянии организма. Однако, радиотехнические расчеты и методы получения диагностической информации выходят за рамки настоящей работы.

Ключевым моментом фрактального моделирования является получение аналитического фрактального выражения для графического отображения фрактала. Из множества фрактальных выражений (геометрические, алгебраические, стохастические) автором были выбраны геометрические, т.к. позволяют быстро и достоверно получить топологию органа или системы органов. Образ фрактала строится на основе L-системы (А. Линденмайер, 1968), для которой автором был разработан скрипт для MatLab 2016 (существуют версии для Windows и Linux). Предварительно, на основе графического изображения органа или системы органов, формируется развернутое L-выражение, которое затем подвергается свертке. Полученные аксиома, продуцирующее правило, правила для вспомогательных переменных X и Y, а также углы альфа и тэта, вводятся в качестве переменных в скрипт. В процессе выполнения скрипта формируется фрактальное графиче-

ское отображение и сохраняются координаты каждой из точек фрактала, которые используются для дальнейших расчетов ЭМИ организма. Предлагаемая методика может использоваться не только в биологии и медицине — диапазон применения может быть расширен практически на все предметные области, объекты которых содержат фракталы. В связи с этим, автор считает, что на основе представленного материала может читаться курс по фрактальному моделированию для студентов старших курсов естественно-научных факультетов университетов, включая биологию и медицину.

С. С. Дереченник, С. В. Лапич, М. В. Николаюк-Ртищева
Брест, Брестский государственный технический университет

Моделирования шумовых сигналов в Octave

Аннотация

Программная модель направлена на определение вероятностных характеристик Гауссова шума и шума Релея. Проект построен на базе GNU Octave. Основными областями применения проекта является микроэлектроника и радиотехника.

В настоящее время большой интерес представляют методы передачи, в которых в качестве носителей информации используются не гармонические колебания, а шумовые сигналы. Особенностью такого вида переносчика информации, как шум, объясняется новыми открывающимися перспективами по организации защиты передаваемых данных на первом уровне эталонной модели OSI [1].

Шум является случайным сигналом и присутствует во всех частотах. Наиболее распространенными моделями случайных сигналов и помех являются телеграфный сигнал, «белый» шум, гауссов шум, шум Релея. В практических задачах радиотехники, в техники связи и в других отраслях, в том числе и в информатике присутствует, как правило, «белый» шум или гауссов шум, который возникает при суммировании статистически независимых «белых» шумов [2]. «Белый» шум является стационарным случайным процессом q(t), у которого автокорреляционная функция описывается дельта-функцией Дирака, а спектральная плотность мощности не зависит от частоты и имеет постоянное значение $G_q(w) = \sigma^2$, равное дисперсии значений q(t).

Модель Гауссов шум («белый» шум) можно отнести к модели AP, которую можно описать разностным уравнением вида: $F_t = \sum_{i=0}^{n}(a * x_{t-i} - \sigma)$, где a — параметр модели AP, x_{t-i} — случайная величина, σ — дисперсия [3].

Рынок программного обеспечения для моделирования шумовых сигналов моделей довольно разнообразен. Однако большая часть используемого программного обеспечения является недоступной обычному пользователю по экономическим причинам. В результате подобные задачи часто приходится решать с применением универсальных языков программирования. Эта причина и послужила основным мотивом для выбора среды разработки данного проекта.

Модель Гауссова шума реализована с помощью мощного пакета GNU Octave, который является свободным ПО, распространяющимся под лицензией GNU GPL [4]. Кроме того, Octave представляет собой интерактивный командный интерфейс для решения линейных и нелинейных математических задач, а также предназначен для проведения численных экспериментов. Используя встроенные функции randn и plot(), построен Гауссов шум, который представляет собой массив случайных величин, и построена гистограмма плотности распределения с помощью команды hist(). Гауссов случайный процесс типа «белый» шум и гистограмма плотности распределения представлены на рисунке 1. Аналогично создана модель шума Релея и построена гистограмма плотности распределения, которые представлены на рисунке 2. Код доступен по адресу `https://github.com/svlapich/signal`.

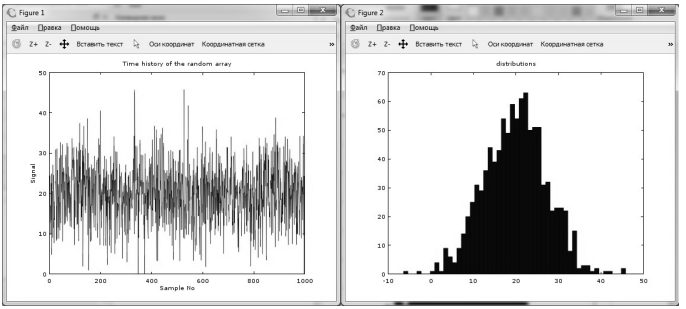


Рис. 1: Гауссов случайный процесс типа «белый» шум и гистограмма плотности распределения (Number of samples — 1000, Give the average — 20, Give the variance — 50)

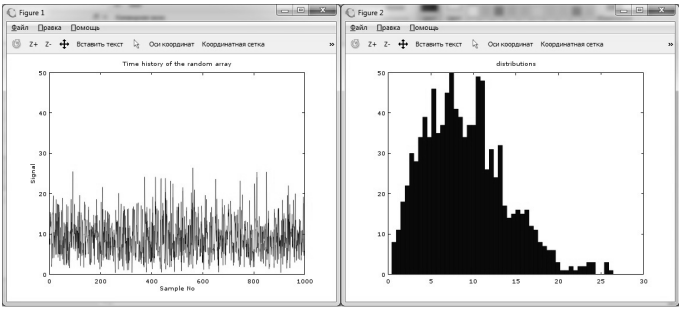


Рис. 2: Шум Релея и гистограмма плотности распределения (Number of samples — 1000, Give the variance — 50)

Ожидаемая область применения проекта связана, прежде всего, с микроэлектроникой и радиотехникой. Например, такая разработка может быть использована при создании многоканальных шумовых модулей с источником случайного сигнала в виде твердотельных диодов-генераторов.

Литература

[1] Корчинский В.В. Модель шумового сигнала для передачи конфиденциальной информации // Вестник НТУ «ХПИ», 2013. — № 11. — С. 89–94.

[2] В.С. Разумейчик, В.В. Буслюк, С.С. Дереченник, В.И. Поляков, С.В. Лапич Оценка вероятностных характеристик случайных сигналов микроэлектронного шумового модуля // Вестник Брестского государственного технического университета, 2014. — № 5 : Физика, математика, информатика. — С. 41–45.

[3] Громаков Ю.А. Структура TDMA кадров и формирование сигналов в стандарте GSM // Электросвязь, 1993. — №10. — С. 9–12.

[4] Алексеев Е.Р., Чеснокова О.В. GNU Octave для студентов и преподавателей. — Донецк.: ДонНТУ, Технопарк ДонТНУ УНИТЕХ, 2011. — 332с.

Сергей Мартишин, Марина Храпченко
Москва, Институт системного программирования РАН

Облачные вычисления над конфиденциальными данными с использованием СПО

Аннотация

Анализируются принципы защиты конфиденциальной информации в NoSQL («не только SQL») системах управления базами данных (СУБД) свободного и свободно распространяемого программного обеспечения (СПО) для выполнения облачных вычислений. Показана необходимость понимания студентами всего комплекса проблем при работе с конфиденциальными данными[1].

В настоящее время объемы информации таковы, что для ее хранения и обработки необходимо использование современных технологий, среди которых наиболее широкое распространение получили облачные вычисления. Необходимо также учитывать, что обработке подлежат в том числе и конфиденциальные данные, требующие специальных мер по их защите. Именно поэтому для студентов и магистрантов, обучающихся по направлениям «Информационные системы и технологии», «Информатика и вычислительная техника», «Программная инженерия» помимо изучения облачных технологий, следует обратить отдельное внимание на аспекты связанные с защитой информации.

В последние годы для хранения такого рода информации используются NoSQL СПО СУБД, а для манипуляции с данными облачные сервисы. Заметим, что основное предназначение облачных вычислений — распределенная обработка данных, то есть в основном решение трудоемких вычислительных задач над большими объемами информации, зачастую неструктурированной. Такого рода информация возникает в различных сферах. Это могут быть данные с устройств видеорегистрации, из социальных сетей, связанные с научными исследованиями. Заметим, что конфиденциальная информация, обработка которой имеет свои жесткие требования, чаще всего возникает

[1]Работа выполнена при финансовой поддержке РФФИ, проект № 16-01-00714

при обработке большого массива информации, например, медицинские данные, социологические исследования и пр.

Для выполнения различных задач облако предоставляет пользователю ресурсы: процессорное время, оперативная память, сетевые каналы, программное обеспечение и т.д., как правило, в виде одной из моделей обслуживания: IaaS (инфраструктура как услуга), PaaS (платформа как услуга) или SaaS (программное обеспечение как услуга). Некоторое время назад появилась модель «память как сервис». Однако вне зависимости от выбранной модели обслуживания пользователю необходимо решить вопросы безопасности. Особенно этот аспект является важным в случае работы с конфиденциальными данными, поскольку их хранение и обработка происходит на облаке, которое является противником (в самом простом случае выполняющим запросы по протоколу, но анализирующим информацию, пытаясь узнать конфиденциальные данные).

Как было сказано выше, для организации хранения неструктурированных данных традиционные реляционные СУБД с вертикальной масштабируемостью и регулярной структурой не эффективны, в отличие от горизонтально масштабируемых (с возможностью размещения на нескольких независимых серверах) NoSQL СУБД. Наиболее популярными в настоящее время являются СПО типа «ключ-значение» и документные (например, Redis, MongoDB, CouchDB и т.п.) [1].

Одним из наиболее удобных инструментов для создания приложений, обрабатывающих большие объемы данных, является серверная технология Node.js. Язык JavaScript, используемый и на стороне сервера в платформе Node, позволяет разработчикам создавать веб-приложения для клиентской и серверной части на одном языке. Особенно важно, что широко распространенный формат обмена данными JSON (применяется также в различных базах данных NoSQL, например, в MongoDB, CouchDB) является собственным форматом JavaScript. Заметим, что Node.js хорошо подходит для распределенной обработки информации на облаке.

В случае распределенного хранения и обработки наиважнейшим становится вопрос информационной безопасности. Традиционные встроенные средства защиты СУБД, Node.js и самой модели обслуживания облака, такие как: аутентификация и контроль прав доступа пользователей, использование брандмауэра, VPN, хеширование паролей и пр., решают задачу лишь частично. Казалось бы, для защиты

конфиденциальных данных, размещаемых и/или вычисляемых на облаке (в случае, когда пользователь не доверяет, облаку) достаточно применить традиционные криптографические средства.

Однако возникает вопрос о том, какие запросы могут быть разрешены для клиентов, которые могут являться потенциальными пассивными противниками (случай активного противника требует значительно более серьезных способов защиты). Например, если интересующий противника элемент данных является целым числом из известного диапазона, то хорошо известным методом деления пополам значение этого элемента будет определено вне зависимости от используемой системы защиты информации, в том числе гомоморфной криптосистемы [2,3]. Подобную угрозу можно пытаться предотвратить, вводя запреты на некоторые типы запросов. Причем запреты должны вводиться вне зависимости от используемого СПО. В работе [4] исследована дедуктивная безопасность запросов к базам конфиденциальных данных для некоторого класса функций. Использование этих функций позволяет гарантировать стойкость предложенной гомоморфной криптосистемы.

Таким образом помимо изучения СПО для создания облачных приложений (Node.js, MongoDB и пр.) и встроенных в данное СПО средств защиты информации, очень важно понимание студентами всего комплекса проблем, связанного с распределенной обработкой конфиденциальных данных в облачных вычислениях, а реализация защиты данных в распределенных приложениях должна осуществляться разработчиком с учетом вышеизложенных принципов и опираться на последние исследования в области защиты информации.

Литература

[1] *Мартишин С. А., Симонов В. Л., Храпченко М. В.* Базы данных. Практическое применение СУБД SQL и NoSQL типа для проектирования информационных систем: учебное пособие, М: ИД Форум — Инфра-М, 2016, 368 с, — ил. — (Высшее образование) ISBN 978-5-8199-0660-6 (ИД «ФОРУМ»), ISBN 978-5-16012141-3 (ИНФРА-М, print), ISBN 978-5-16-104936-5 (ИНФРА-М, online).

[2] *Варновский Н. П., Мартишин С. А., Храпченко М. В., Шокуров А. В.* Пороговые системы гомоморфного шифрования и защита информации в облачных вычислениях. // «Программирование», — 2015. № 4 — с. 47–51.

[3] *Варновский Н. П., Захаров В. А., Шокуров А. В.* К вопросу о существовании доказуемо стойких систем облачных вычислений // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. — 2016. — № 2. — С. 32–38.

[4] *Варновский Н. П., Захаров В. А., Шокуров А. В.* О дедуктивной безопасности запросов к базам конфиденциальных данных в системе облачных вычислений // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. — 2017. — № 1. (в печати).

Александр Крюков
г. Рязань, Рязанское высшее воздушно-десантное командное училище имени генерала армии В.Ф.Маргелова

Задача выбора программного обеспечения для учебного процесса

Аннотация

Свободное программное обеспечение позволяет использовать устаревшее оборудование до полного износа и парировать ряд угроз безопасности.

Дано:

1. Требования федеральных государственных образовательных стандартов высшего образования — 11.05.04 и среднего образования — 11.02.09-11.02.11 к компетенциям и уровням знаний, умений и навыков выпускников по общепрофессиональным дисциплинам.

2. Широкий спектр уровней подготовки обучающихся — от 1 курса получающих среднее образование до проходящих переподготовку специалистов с высшим образованием.

3. В классе проводятся групповые, практические, лабораторные занятия, дипломное проектирование.

4. Компьютерный класс используется для самостоятельной работы обучающихся, при этом преподавательский и лаборантский состав в классе может отсутствовать.

5. Шаблоны разрабатываемых документов «заточены» под MS Office.

6. Ряд унаследованных программ работают под Windows.

7. Источник баз антивирусного программного обеспечения опаздывает в среднем на неделю. Заносы свежих вирусов на носителях пользователей происходят регулярно.

8. Время преодоления административных барьеров для доступа в интернет превышает время работы в интернете.

9. Материальная база устарела и изношена. На материнских платах большей части компьютеров класса отсутствуют порты SATA.

Найти: программное обеспечение для учебного процесса.

Решение.

1. Поскольку жёсткие диски с интерфейсами IDE больше не производятся, замена дисков с ошибками позиционирования невозможна.

2. Ошибки чтения скопированных на диск файлов дистрибутива не позволяют установить на диски с ошибками Windows.

3. Последовательным подключением исправного диска с установленным свободным программным обеспечением (ПО) к разнотипным компьютерам класса выбираются дистрибутивы, загружающийся на большинстве из них.

4. Из числа проверенных дистрибутивов выбираются те, на которые могут быть установлены унаследованные программы.

5. На жёсткий диск без ошибок компьютера, подключенного к Интернет, устанавливается выбранный дистрибутив и производится его обновление.

6. Добавляются пользователи с различными правами — один или два, как правило, с правами пользователя, не входящего в группы root и sudo.

7. Устанавливаются программы, необходимые для учебного процесса. Программы, работающие под Windows, ставятся на wine. В папки пользователей копируются необходимые файлы.

8. На компьютеры класса программное обеспечение не устанавливается, а клонируется на IDE диски с подготовленного диска, не имеющего ошибок.

9. Компьютеры тестируются на работоспособность всего установленного ПО, выполняются необходимые настройки.

10. Как правило, по-умолчанию компьютер загружается с автоматическим вводом пароля пользователя, не входящего в группы root и sudo, которому недоступны программы, работающие под wine, а также установка программ. Пользователь может читать учебники, работать с файлами, копировать их на свои носители и обратно, выполнять пропущенные лабораторные работы и тесты в режиме обучения,

играть в небольшое количество игр, устанавливаемых в системе по-умолчанию. Подключить модем или телефон в режиме модема можно, но выйти в Интернет не получится, так как необходимое для этого ПО не установлено. Зараженный носитель подключить можно, он примонтируется, «невидимые» и «зашифрованные» файлы и папки будут в файловом менеджере видны также, как тело вируса. Вирус скопировать в папку пользователя можно, он будет виден, но запустить его не удастся, так как не хватит прав.

11. Дипломникам и проходящим переподготовку при необходимости выдаются пароли пользователей, которым доступны программы, установленные на wine. Они расписываются за их сохранность и сохранность ПО. Их предупреждают, что примонтированные носители сначала нужно открыть в файловом менеджере и убедиться, что неизвестных исполняемых файлов на них нет, скопировать файлы, необходимые для работы, в папку, отмонтировать и отключить носитель, и лишь потом запускать программы на wine. По окончании работы сначала останавливается wine, а потом переработанный файл копируется на носитель.

12. Установка антивирусного ПО на основную операционную систему или wine возможна, но, поскольку оно занимает оперативную память, которой недостаточно, и загружает изношенную дисковую систему — не целесообразна. Компьютеры в классе ломаются регулярно, но за 2 года вирус ни разу не стал причиной выхода их из строя.

13. Какое-то время компьютеры с «убитыми» дисками работают, а потом работа нарушается. Тогда операция клонирования ПО повторяется — до тех пор, пока диск не перестанет читаться.

14. В классе установлены операционные системы Debian v7.7, Runtu XFCE 14.4, программы LibreOffice 4.2, Scilab 5.5.

Евгений Алексеев
Киров, Вятский Государственный Университет

Свободное программное обеспечение при подготовке бакалавров и магистров на кафедре фундаментальной информатики и прикладной математики Вятского Государственного Университета

Аннотация

Представлен опыт использования свободного программного обеспечения на кафедре фундаментальной информатики и прикладной математики Вятского Государственного Университета.

При обучении бакалавров и магистров направления «Прикладная математика и информатика» на кафедре фундаментальной информатики и прикладной математики (ФИиПМ) Вятского государственного университета (ВятГУ) в последние три года достаточно широко используется свободное программное обеспечение. Несколько курсов в основном построены на свободном программном обеспечении по управлением ОС семейства Linux:

1. Операционные системы.

   Классический теоретический курс: история развития операционных систем; общие сведения об операционных системах; процессы, алгоритмы управления процессами; управление памятью;файловая система современного компьютера.

   Практические и лабораторные занятия: файловая система unix-подобных операционных систем; установка ОС семейства Linux на компьютер, первоначальная настройка рабочего стола; команды терминала Linux; знакомство с репозиторием программного обеспечения (ПО), установка программ в ОС Linux; управление пользователями в Linux; средства создания загрузочной флешки; компиляция и отладка программ в ОС Linux (знакомство с компилятором gcc, отладчик gdb); программирование алгоритмов управления процессами; утилиты сборки дистрибутива.

   Курсовой проект: сборка специализированного дистрибутива.

2. Параллельные вычисления. Курс предназначен для знакомства студентов третьего курса с параллельным программированием. Лабораторные работы проводятся в лабораториях кафедры ФИиПМ на компьютерах под управлением ОС Lubuntu. Используются компиляторы gcc, g++, gfortran, текстовый редактор Geany, библиотека MPICH. Работа организована как на локальных ПК, так и на вычислительном кластере ВятГУ. Изучается язык программирования Фортран — как язык разработки параллельных и высокоэффективных вычислений. Студенты знакомятся с технологиями параллельного программирования MPI и OpenMP (языки программирования C и Фортран). Разрабатывают параллельные MPI-программы для решения классических задач вычислительной математики.

3. Администрирование в вычислительных сетях. Курс полностью построен на базе ОС семейства Linux.

4. Архитектура параллельных вычислительных систем. Один из первых курсов при подготовке магистров (профиль «Технологии параллельного программирования и высокопроизводительные вычисления»). Кроме изучения теоретических вопросов архитектуры современных вычислительных систем, магистранты в лаборатории математического моделирования кафедры ФИиПМ самостоятельно «строят» локальный вычислительный кластер. Локальный вычислительный кластер и вычислительный кластер ВятГУ используется для проведения лабораторных работ по курсу.

5. Технологии параллельных вычислений. Магистранты изучают параллельное программирование с использованием технологий CUDA и OpenMP.

6. Параллельные численные методы. Завершающий курс по параллельным вычислениям в магистратуре, в котором магистранты с использованием технологии параллельного программирования разрабатывают параллельные приложения решения задач вычислительной математики с использованием технологий MPI, OpenMP, Co-array на языках C и Фортран. Используются свободно-распространяемые средства разработки и компиляторы компании Intel. Курс заканчивается курсовым проектом и экзаменом.

Обучение параллельным вычислениям проводится на базе лаборатории математического моделирования и вычислительного кластера ВятГУ. В лаборатории установлена Ubuntu 14.04 с необходимым ПО (g++, gfortran, Mpich, geany, компиляторы Intel), на базе лаборатории в осеннем семестре 2016-17 уч. года построен локальный вычислительный кластер. Начато проведение лабораторных работ на его базе. Кафедре ФИиПМ на вычислительном кластере ВятГУ выделено 16 вычислительных узлов, на которых развёрнута операционная система Ubuntu 14.04 с необходимым ПО.

Свободное программное обеспечение также широко используется при подготовке дипломов бакалаврами и написании магистерских диссертаций на кафедре ФИиПМ ВятГУ. В прошлом учебном году была написана магистерская диссертация «Анализ быстродействия параллельных программ при решении актуальных задач вычислительной математики». В этом году планируется защитами бакалаврами выпускных работ «Сравнительный анализ свободных и проприетарных компиляторов языка Фортран при решении вычислительных задач», «Кроссплатформенный программный интерфейс на языке C++ построения графиков функций», «Разработка кроссплатформенной библиотеки на языке Фортран построения графиков функций» и магистерской диссертации «Использование современных технологий параллелизма в языках Фортран и C(C++) при реализации алгоритмов вычислительной математики».

Николай Власенко
Киев
http://forums.wesnoth.org/viewtopic.php?f=19&t=43919

Разработка дополнений к игре Wesnoth

Аннотация

Wesnoth открытая игра. Играть в нее достаточно интересно, но еще интереснее самому создавать к ней расширения (и это вполне легко).

Wesnoth — это игра с открытыми исходниками. Расширения для этой игры тоже открытые. Любой, кто хочет, может написать свой add-on и выложить его на общедоступный сервер add-on-ов для wesnoth. Мое увлечение началось три с половиной года назад, когда мой папа показал, как устроена игра внутри, и мы с ним начали

писать свою первую кампанию и создали несколько сценариев. К сожалению, та кампания так и осталась неоконченной (и мы случайно ее потеряли), но с тех пор мне стало интереснее заниматься хакерством существующих add-on-ов (имеются в виду локальные копии) — менять свойства юнитов и события, добавлять юнитов из других аддонов и смотреть, как add-on-ы устроены внутри. В какой-то момент мне очень понравилось, как в кампании для wesnoth 1.8 под названием «The Foolish Hero» была по-своему реализована система Inventory. Но тогда вышел wesnoth 1.10, с которым функция Inventory в «The Foolish Hero» была не совместимой из-за изменений в WML, и я захотел починить Inventory. В итоге я стал писать свою первую полностью самостоятельную кампанию «Tale of Alan», которую в итоге выложил на общедоступный сервер add-on-ов для wesnoth. Кампания «Tale of Alan» собрала больше двух тысяч скачиваний под wesnoth 1.10. Потом я ее переписал для wesnoth 1.12, где она собрала еще больше тысячи скачиваний.