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

Одиннадцатая конференция «Свободное программное обеспечение в высшей школе»

Переславль, 30–31 января 2016 года

Сборник материалов конференции

Москва,
Альт Линукс,
2016

Одиннадцатая конференция «Свободное программное обеспечение в высшей школе»: Материалы конференции / Переславль, 30–31 января 2016 года. М.: Альт Линукс, 2016. — 120 с. : ил.

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

ISBN 978-5-905167-20-1

Программа конференции

30 января

11.30-12.30 Регистрация участников и заселение

Дневное заседание
12.30–14.20

12.30 А. Е. Новодворский. Открытие. Информация оргкомитета

12.40 А. В. Смирнов. Свободное ПО и преференции для отечественного ПО при госзакупках

13.00–13.40 Н. Н. Непейвода, проф. и др.

Самое высшее образование: проект edverest 7

13.40–14.00 А. А. Рыжов, Л. Н. Чернышов

Курс «Компьютерная лингвистика» 9

14.00–14.20 Б. А. Новиков

Предподавание технологий СУБД на основе PostgreSQL ... 11

14.20–15.40 Перерыв на обед

Вечернее заседание
15.40–18.10

15.40–16.00 Е. А. Чичкарев и др.

Опыт разработки и внедрения информационной системы учета научной работы преподавателей в высшем учебном заведении 14

16.00–16.20 Н. В. Потехин

Использование свободной СДО ILIAS в учреждении дополнительного профессионального образования 15

31 января

Утреннее заседание
10.00–13.10

Дневное заседание
14.30–17.50

Вне программы

С. М. Абрамов, А. В. Шкред, Н. Н. Непейвода
Переславль-Залесский, ИПС РАН, ИНТУИТ, Университет города Переславля

САМОЕ ВЫСШЕЕ ОБРАЗОВАНИЕ: проект edverest

Замысел проекта

Создание ведущего образовательного интернет-проекта, реализующего современные и перспективные образовательные методики и технологии, предоставляющего эффективные веб-сервисы для организации процесса обучения индивидуальных пользователей и групп, а также массового обучения. В основе архитектуры проекта объединены модели систем дистанционного обучения (СДО, Learning Management Systems — LMS), социальных сообществ (Social Networks), футуристических идей Web 3.0 и активного участия профессионалов.

Особенности проекта

Учебные курсы создаются ведущими преподавателями учебных заведений и специалистами компаний. В основу учебных материалов положены актуальные и перспективные идеи с учетом лучшего мирового опыта, без оглядки на стандарты Минобразования.

Структура всех курсов, в том числе и непрофильных, строится с учетом их использования в предметной области, например, в математических курсах активно используются программные инструменты для демонстрации абстрактных понятий и постоянно устанавливаются взаимосвязи с информатикой, как в виде добрых советов, так и в виде предостережений. Задачи направлены прежде всего на обучение математическому моделированию и разработке наукоемких приложений. Соответственно, в других курсах взаимосвязи тоже устанавливаются, характер задач остается прежним, но сферы их направленности изменяются (вместо «наукоемкие» могут быть «надежные, удобные в использовании» и так далее; вместо математического моделирования. Скажем, модели бизнес-процессов).

Обязательным условием обучения является практика и стажировка в компаниях.

Дипломная работа является условием завершения учебной программы. Тема дипломной работы выбирается студентом самостоятельно. Результатом выполнения дипломной работы является представление и защита готового программного продукта (например, для мобильного устройства принятого к распространению в магазине приложений, для СПО выложенного в репозиторий), интернет-сайта или сервиса, представленного в публичном доступе или программного кода в рамках значимой разработки.

Проект ориентирован на массовое обучение российских и зарубежных информатиков на русском и английском языках.

Принципы формирования учебных программ

Учебные планы и содержания дисциплин строятся исходя из необходимости объема теоретических знаний и практических навыков. Каждый семестр обучения завершается получением сертификата по системе разработки или моделирования (например, по одной из систем работы с базами данных, языку программирования, системе моделирования бизнес-процессов). Для завершения обучения необходимо пройти обязательный минимум учебных курсов с учетом выбранной специализации. Обучающемуся предоставляется широкий выбор возможных вариантов специализаций. Фиксируется, но тоже в трех вариантах (математические модели, инженерия, взаимодействие с пользователем) лишь набор фундаментальных курсов. Образование должно формировать активного специалиста на всю жизнь и быть востребованным на рынке труда немедленно и с учетом перспектив развития отрасли и не только её, например, экономики. Учебный процесс предполагает самостоятельное приобретение знаний и их объективную независимую проверку специалистами. Учебная практика и стажировки в реальных компаниях являются обязательным элементом учебного плана.

Семестры 1–4 предполагают получение основ технологий и фундаментального базиса. Изучать курсы в них можно в любом порядке, с учетом их логической зависимости. После завершения всех курсов этих семестров — обязательная учебная практика сроком не менее 30 дней.

Семестры 5–8 — специализация, курсы внутри семестров изучаются в любом порядке, для завершения необходимо пройти практику сроком не менее 60 дней.

Семестр 9 — продолжение специализации, защита темы дипломной работы, стажировка не менее 30 дней.

Семестр 10 — продолжение специализации, создание проекта дипломной работы и его защита.

Поскольку образование дистанционное, то предполагается, что обучающиеся реально начинают работать на достаточно ранних этапах обучения. Поэтому сроки практики и стажировки условные, они касаются только студентов, учащихся параллельно в других «официальных» образовательных учреждениях.

А. А. Рыжов, Л. Н. Чернышов
Москва, НИУ МАИ, Финансовый университет

Проект: Курс «Компьютерная лингвистика»
http://ucheba8fac.wix.com/linguistics

Курс «Компьютерная лингвистика» на основе библиотеки NLTK

Аннотация

В настоящее время во множестве отечественных университетов преподаются такие курсы, как «Компьютерная лингвистика» и «Основы обработки текстов». Но методика преподавания данных дисциплин полностью зависит от конкретного учебного заведения и не всегда соответствует современным реалиям. В докладе проведен анализ основных учебных программ ведущих университетов по данной дисциплине и разработана типовая структура курса на основе открытой библиотеки NLTK. Разработаны задачи к лабораторному практикуму и описаны возможные направления для научной деятельности студентов.

Задача создания эффективного спецкурса обучения студентов компьютерной лингвистики и текстовому анализу крайне актуальна на данный момент, ввиду потребности в специалистах, владеющих этими предметами, и достаточно большого обилия средств и методик. Анализ учебных курсов, читаемых в таких вузах как МГУ, ВШЭ, МФТИ, НГУ и некоторых других показал, что программы разительно отличаются друг от друга, как по теоретическому материалу, так и по предлагаемым студентам практическим и лабораторным задачам. Все рассмотренные программы относились к направлениям подготовки «Прикладная математика и физика» и «Фундаментальная и прикладная лингвистика».

В докладе представлена типовая структура спецкурса для подготовки студентов на основе лингвистической библиотеки NLTK, который поддерживается созданным интернет-ресурсом [1]. Данная библиотека успешно используется во многих университетах мира в курсах [2], связанных с текстовым анализом и со смежными областями. NLTK — открытая библиотека реализована на языке Python и была создана специально для курса компьютерной лингвистики Университета Пенсильвании [4]. Цель создания собственного интернет-ресурса в том, чтобы преподаватели различных вузов могли использовать представленные наработки для составления своих учебных программ, соответствующих современным реалиям подготовки студентов.

Представлен набор задач и лабораторных работ на основе курса. Общий формат лабораторных не подразумевает создания большого количества задач. Для практикума планируется 1–2 задачи, которые будут покрывать широкое поле основных понятий и терминов. К подобным заданиям можно отнести задачу автоматического определения тональности. В ходе ее решения студент познакомится и на практике столкнется с такими важными понятиями, как алгоритмы классификации, алгоритмы стемминга, текстовые корпуса, N-граммы. Студенту предложено на основе имеющегося или составленого им корпуса определенной предметной области, запрограммировать собственный классификатор и получить точность не ниже определенного порога. Для оценивания результата возможно использовать автоматическую систему проверки.

Стоит заметить, что библиотека NLTK представляет только базовые средства текстового анализа и предполагается, что разработчик на основе данных инструментов, будет создавать свои более высокоуровневые программы. Базовые средства не всегда достаточны для решения разнообразных задач. Незатронутыми остаются такие задачи, как автореферирование и выделение именованных сущностей в тексте. В настоящий момент в данной библиотеке крайне мала поддержка русского языка. С одной стороны, это плохо, но с другой стороны, множество разработчиков постоянно совершенствуют библиотеку. Студентам предоставляется возможность самим поучаствовать над её развитием и предложить свои модули, разработанные в рамках лабораторных работ или в качестве научной работы.

Литература

[1] Курс «Компьютерная лингвистика» URL: `http://ucheba8fac.wix.com/linguistics` (дата обращения: 12.01.2016).

[2] Courses that use NLTK // tinyurl URL: `http://tinyurl.com/nltk-courses` (дата обращения: 10.01.2016).

[3] *Bird S., Klein E., Loper E.* Natural language processing with Python. — "O'Reilly Media, Inc.", 2009.

[4] Исходный код NLTK // github URL: `https://github.com/nltk/nltk` (дата обращения: 10.01.2016).

Борис Асенович Новиков, Иван Евгеньевич Панченко
Санкт-Петербург, Москва, СпБГУ, Постгрес Профессиональный

Предподавание технологий СУБД на основе PostgreSQL

Аннотация

Часто преподавание в ВУЗах предметов, связанных с базами данных, ведется на основе проприетарных СУБД, вендоры которых хорошо понимают, что это важный канал для формирования клиентуры. Однако несправедливо, что ВУЗы используются в этом качестве, их цель — образование. Использование одного продукта снижает кругозор студентов, а закрытость продукта приводит к отсутствию понимания внутреннего устройства СУБД. Поэтому PostgeSQL, как одна из наиболее совершенных СУБД с открытым кодом, является хорошим вариантом для построения на её основе качественного учебного процесса. Б.А. Новиков, профессор СПбГУ, расскажет о своем опыте преподавания СУБД на базе PostgreSQL, а И. Е. Панченко представит новый учебный курс «технологии баз данных» для ВУЗов, разрабатываемый компанией Postgres Professional совместно с СПбГУ.

Требования к специалистам, выпускаемым университетами, должны включать как наличие практических навыков применения программных систем, актуальных на период обучения, так и фундаментальных знаний, которые остаются актуальными в течение десятилетий даже в такой быстро развивающейся области деятельности, как информатика. Применительно к системам управления базами данных

(СУБД) это означает, с одной строны, навыки создания высокоэффективных приложений на базе высокопроизводительных СУБД, с другой стороны, знание моделей, алгоритмов и технологий, лежащих в основе реализаций СУБД.

Цикл дисциплин, предусмотренных учебными планами Санкт-Петербргского государственного университета, включает как дисциплины, в первую очередь ориентированные на овладение навыками использования СУБД, в частности, знание языка запросов SQL, так и дисциплины, посвящённые вопросам внутреннего устройства СУБД, в частности, методам индексирования, обработки запросов и теорию транзакций.

В случае, если выработка навыков использования СУБД ведется исключительно на основе проприетарных СУБД, вендоры которых хорошо понимают, что это важный канал для формирования клиентуры и поэтому предоставляют свои продукты на льготных условиях, возникает разрыв между практическими навыками и теоретическими основами, так как преподавание теоретических дисциплин в этом случае ведется исключительно на основе публикаций.

Поэтому PostgeSQL, как одна из наиболее совершенных СУБД с открытым кодом, является хорошим вариантом для построения на её основе качественного учебного процесса. Фактически это единственная система с открытым кодом. которая может быть отнесена к классу высокопроизводительных и поэтому навыки её использования могут непосредственно использоваться выпускниками на рабочих местах. С другой стороны, открытость даёт возможность устранения разрыва между преподаванием теории и используемыми реализациями.

Широкие возможности доступа к программному коду дают условия как для пассивного ознакомления с внутренним устройством системы, так и для экспериментов с собственными реализациями отдельных алгоритмов, как в учебных заданиях, так и в исследовательских проектах. Практика такого использования системы широко распространена в исследовательских группах в университетах всего мира. Конечно, обычно такие исследовательские проекты не приводят к созданию полноценных расширений PostgeSQL.

Новый создаваемый авторами учебный курс «Технологии баз данных» содержит как практическую, так и теоретическую информацию, позволяющую после его освоения не только эффективно использовать СУБД, пользуясь знаниями об её внутреннем устройстве и алгоритмах, но и участвовать в разработке новых программных продуктов,

использующих технологии СУБД, включая современные алгоритмы поиска данных, организацию обработки транзакций, работу в распределенных системах.

Курс рассчитан на студентов младших курсов (бакалавриата) классических и технических университетов, а также других ВУЗов, имеющих базовую подготовку по программированию и продолжающих специализироваться в областях, близких к программированию.

В курсе подробно рассматриваются основные понятия, устройство и принципы работы СУБД, а также технологии (архитектура, алгоритмы, структуры данных) лежащие в их основе. Прослушавшие курс получат уверенные знания и практические навыки по следующим вопросам: устройство и принципы работы СУБД; проектирование баз данных; работа с SQL — составление и оптимизация запросов; разработка серверных приложений; использования различных типов индексов; обработка транзакций и одновременный доступ; основы эксплуатации баз данных; обеспечение надежности хранения, отказоустойчивости и высокой доступности; принципы организации и работы параллельных и распределенных СУБД; работа со слабоструктурированными данными (json, XML). Такая подготовка позволит на старших курсах (в магистратуре) специализироваться на разработке и настройке приложений баз данных, либо в областях проектирования и разработки СУБД.

Курс состоит из двух частей. В первой части рассматриваются базовые сведения о базах данных и системах управления базами данных: реляционная модель данных, язык SQL, обработка транзакций.

Во второй части подробно рассматриваются технологии, лежащие в основе функционирования СУБД, а также современные направления и тенденции развития СУБД, и основные аспекты их практического применения. При этом некоторые темы, рассмотренные в первой части, изучаются повторно на более глубоком уровне. Курс в основном касается классических реляционных и объектно-реляционных СУБД, но затрагивает также тематику неклассических СУБД.

Практические занятия не только помогают закрепить пройденный на лекции материал. Они содержат много дополнительной информации,закрепляющей и расширяющей знания, изложенные в теоретической части. В качестве СУБД для практических занятий используется PostgreSQL. Как первая, так и вторая часть курса может быть выделена в самостоятельный курс. Отдельные разделы курса могут быть скомбинированы так, чтобы получить курс более практической

или более фундаментальной направленности, или адаптировать курс к конкретному учебному плану ВУЗа.

Чичкарев Е. А., Сергиенко А. В., Дьячук М., Анисимов С.
г. Мариуполь, Украина, ГВУЗ «Приазовский государственный техническийуниверситет»
http://pstu.edu

Опыт разработки и внедрения информационной системы учета научной работы преподавателей в высшем учебном заведении

Аннотация

Проведен анализ существующих систем автоматизации работы ВУЗов, а такжеиспользуемых технологических решений.Разработана и проходит опытную эксплуатацию информационная система учетарезультатов научной работы преподавателей высшего учебного заведения.Система позволяет вести учет публикаций, диссертаций, отчетов, ведущихсянаучных работ и т.п., а также на основании балльной оценки всех видовработ подсчитывать общий результат работы сотрудника или подразделения.Система реализована на PHP+MySQL с использованием фреймворка Kohana.

В рамках работы, направленной на совершенствование управления и качества работы сотрудников Приазовского государственного технического университета, разработана информационно-аналитическая система мониторинга и учета результатов научной деятельности.

Актуальность выполнения настоящей работы обусловлена пересмотром подходов к организации работы профессорско-преподавательского состава ВУЗов, сокращенем аудиторной нагрузки на преподавателя и повышением требований Министерства образования к результатам научной работы ВУЗов. Аналитическая информация и плновые показатели соответствующего характера постоянно требуется в текущей деятельности университета: для формирования отчетов структурных подразделений по результатам научной деятельности, написания заявок на участие в проектах и контроля выполнения научной работы. В частности, накапливаемая информация о публикациях, методической работе, выполнении НИР силами преподавателей

и сотрудников используется для построения рейтингов профессорско-преподавательского состава университета.

Информационно-аналитическая система реализована полностью с использованием открытых программных средств (использован веб-фреймворк Kohana, СУБД MySQL, средства разработки — преимущественно NetBeans).

Система включает модуль учета публикаций и прочих работ (учебных пособий, учебников и т. п.), научно-исследовательских работ, патентов, а также модуль планирования показателей работы сотрудников и подразделений университета.

Перечень типов публикаций является гибким и может изменяться администратором системы. Предусмотрена также гибкая система свойств для различных типов учитываемых публикаций и видов работ.

Для реализации информационно-аналитической системы были использованы современные методики построения программного обеспечения на основе шаблона проектирования MVC (Модель — Представление — Контроллер), а также самые актуальные информационные технологии разработки веб-приложений. Информационная система выполнена в виде веб-приложения, имеет модульную архитектуру. Серверная часть выполняется на платформе Apache-PHP-MySQL, веб-интерфейс является основным способом доступа конечных пользователей к функционалу системы. Предусмотрен вывод нескольких видов отчетов как на веб-страницы, так и в виде готового документа.

Потехин Николай Владимирович
Ярославль, ГАУ ДПО ЯО «Институт развития образования»
http://ilias.iro.yar.ru/

Использование свободной СДО ILIAS в учреждении дополнительного профессионального образования

В методических рекомендациях по реализации дополнительных профессиональных программ говорится о том, что организации, осуществляющие образовательную деятельность, в соответствии с положением Федерального закона от 29 декабря 2012 г. № 273-ФЗ «Об образовании в РФ» вправе при реализации образовательных программ

использовать дистанционные образовательные технологии при всех формах получения образования формы. Применение электронного обучения, дистанционных образовательных технологий основано на положениях Федерального закона № 273-ФЗ и рядом других нормативных актов. При реализации образовательных программ с использованием только дистанционных образовательных технологий (далее — ДОТ) требуется создать условия для функционирования единой информационно-образовательной среды, включающей в себя электронные информационные и образовательные ресурсы, совокупность информационных и телекоммуникационных технологий, соответствующих технологических средств и обеспечивающую освоение обучающимися образовательных программ.

С 2013 года сотрудниками Центра информационных технологий ГАУ ДПО ЯО ИРО велась работа по изучению различных систем дистанционного обучения (далее — СДО). На этом этапе были рассмотрены следующие СДО (ATutor `http://www.atutor.ca/atutor/`, Claroline `http://www.claroline.net/`, Dokeos `http://www.dokeos.com/`, LAMS `https://www.lamsfoundation.org/`, Sakai `https://sakaiproject.org/`, ILIAS `https://www.ilias.de/`, OLAT `http://www.olat.org/`). Был проведен анализ по различным параметрам, в частности по совместимости со стандартом SCORM, администрированию системы, возможностями расширения.

В результате анализа и практической проверки была выбрана система ILIAS. Она имеет инструменты экспорта наработанного материала в стандартизированные форматы, которые затем успешно импортируются существующими системами дистанционного обучения. Широкий базовый функционал, поддержка стандартов, возможность настройки учебного процесса. Имеет большую и достаточно подробную документацию. Данная система используется для дистанционного обучения в ФГОУ высшего профессионального образования «Амурский государственный университет» `http://ilias4.amursu.ru/`.

На этапе внедрения решались следующие задачи: разработка нормативных документов, обучение сотрудников, технический запуск системы.

Были разработаны локальные нормативные документы: положение об использовании ДОТ при реализации дополнительных профессиональных программ, регламент подготовки и размещения курсов для проведения обучения с использованием СДО, требования к структуре и оформлению учебно-методических комплектов, инструкции

для пользователей (преподавателя, слушателя) курсов с использованием СДО.

Технический запуск системы включал в себя установку программного обеспечения необходимого для запуска СДО, её настройка в соответствии с нормативными требованиями, разработка форм представления материалов для курсов.

Для обучения сотрудников ИРО, ответственных за разработку курсов и проведения обучения с использованием дистанционных образовательных технологий, проведен ряд внутрифирменных семинаров. На семинарах были представлены разработанные локальные нормативные документы, проведено обучение работе в СДО, даны рекомендации профессорско-преподавательскому составу ИРО по разработке курсов с использованием СДО. Созданы обучающие/тестовые курсы для проведения обучения с использованием СДО.

С 2014 года с использованием СДО было обучено более 1600 слушателей, разработано 9 курсов повышения квалификации, создано 2 интернет-сообщества педагогов Ярославской области.

За время эксплуатации СДО выявился ряд проблем: недостаточная ИКТ-компетентность и преподавателей и слушателей, сложность в переложении очного курса в дистанционный.

В дальнейшем планируется расширение инструментария дистанционных курсов за счет включения ранее неиспользуемых встроенных инструментов, подключение внешних модулей и интеграция с сервисом видеоконференций.

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

Об эффективности использования метафоры ленточного интерфейса

Аннотация

Выполнено сравнение метафоры ленточного интерфейса (MFI) с типовыми инструментальными панелями FOSS-аналогов для интерфейсов со средней (графические редакторы) и с повышенной (офисные приложения) плотностью виджетов. Оценивается эффективность работы оператора на основе измерения скорости выполнения заданий,

мониторинга физической и умственной нагрузки на примере студентов IT специальностей, знакомых с обеими парадигмами интерфейса. Обсуждаются условия, при которых приложения с MFI показывают в среднем более высокие результаты в тестах. Анализируется возможность применения в Linux-системах ленточной метафоры: обсуждения в сообществе, существующие макеты, аспекты лицензирования, проприетарные MFI-приложения.

Введение

С момента появления «риббонов» (Ribbon, Microsoft Fluent Interface или MFI) в 2007 г. ленточная метафора успела претерпеть эволюционные изменения. Её воплощение в рабочих прототипах предполагало размещение виджетов вверху на прокручиваемой горизонтальной ленте, однако к релизу MS Office прокрутка была заменена вкладками, что и сформировало нынешнее понимание ленточного интерфейса: несколько вкладок, каждая из которых содержит панель с кнопками и, возможно, некоторыми другими виджетами, при том, что виджеты размещены в несколько рядов (с вертикальным разграничением функциональных групп) и различаются по размеру. Вслед за MS Office большинство продуктов при реализации своих ленточных интерфейсов отказались для экономии вертикального пространства от традиционных меню; некоторые приложения реализуют и более сложное поведение MFI — например, миграцию виджетов между видимой и скрытой частями панели, аналогично сокрытию неиспользуемых пунктов меню.

Многократно упоминавшаяся противоречивость ленточного интерфейса определяется двумя дополнительными проблемами, заложенными в его концепции.

Во-первых, это дополнительная визуальная нагрузка из-за большого числа одновременно видимых элементов. Подпункты меню скрыты большую часть времени; в инструментальных панелях с виджетами в одну линию срабатывает избирательность восприятия, выработанная человеком при построчном чтении (к тому же сложных приложениях видимость отдельных панелей можно включать и отключать). MFI «обрушивает» на пользователя всё богатство функционала прямоугольными блоками, и при поиске нужного виджета приходится переключаться с горизонтального сканирования на вертикальное и обратно.

Во-вторых, различные размеры виджетов в пределах одной секции вкладки плохо согласуются с моделью периферического зрения [1], согласно которой человек естественней воспринимает элементы с меньшей детализацией, когда они располагаются у края окна, а элементы с большей детализацией — ближе к центру (в противоположность «прыгающим» размерам пиктограмм в панели MFI, вызывающим ощущение хаоса).

Теоретически у MFI существуют и достоинства — например, возможна более лёгкая обнаруживаемость функционала (discoverability). Однако самый популярный довод в пользу ленточного интерфейса — то, что к нему можно быстро привыкнуть. И, хотя это трудно назвать преимуществом в чистом смысле слова, опросы показывают неплохой уровень удовлетворенности пользователей MFI (как минимум в случае офисного пакета).

Последний фактор послужил для нас побудительным мотивом выполнить практическое сравнение эффективности работы пользователя с MFI и традиционными интерфейсами.

Особенности тестирования

В исследовании приняло участие около 30 студентов IT-специальностей в возрасте 20–22 лет, т. е. пользователи с хорошим опытом работы как с MFI, так и с традиционными панелями. Тестирование выполнялось отдельно для приложений с интерфейсом, перегруженным виджетами, и для приложений со средней визуальной загрузкой панелей. В качестве типовых интерфейсов для тестирования были отобраны три варианта:

- MFI (высоконагруженный интерфейс — приложения пакета MS Office 2007, средненагруженный — Paint из Windows 7);
- классическая верхняя панель (LibreOffice);
- боковая панель (высоконагруженный интерфейс — боковая панель LibreOffice 5.x с отключенной верхней панелью, средненагруженный — свободный графический редактор Pinta, реализующий типовые операции аналогично MFI-прототипу).

В тестовые задания входили операции форматирования и разметки текста, электронной таблицы, либо рисование заданных фигур и примитивных изображений. Тестирование включало сначала выпол-

нение ознакомительных заданий во всех приложениях, а затем других вариантов заданий в режиме мониторинга.

Мониторинг состояния пользователя выполнялся в рамках проекта по разработке программно-аппаратного комплекса [2], который позволяет измерять частоту сердечных сокращений (ЧСС) и электрическую проводимость кожи (ЭПК). В качестве информативного параметра ЭПК оценивалась высокочастотная «фазическая» составляющая кожно-гальванической реакции (КГР) в виде кратковременных импульсов, возникающих в ответ на внешние стимулы, тревогу, напряжение и т. д. Поскольку на результаты измерений ЭПК заметно влияют посторонние факторы, она регистрировалась совместно с ЧСС, что позволяло оценить в комплексе физическую и эмоциональную нагрузку. Наконец, поскольку работа с панелями вносит не такой большой вклад в нагрузку, как работа с документом, выполнялось нормирование ЧСС и КГР по скорости работы для каждого тестируемого относительно его темпа работы с MFI, чтобы уменьшить вклад темпа работы в результаты измерений.

Дополнительно к измерению ЭПК и ЧСС использовался бытовой энцефалограф Neurosky Mindwave, для оценки концентрации внимания пользователя с помощью интегрального параметра, связанного с β-ритмами головного мозга [3].

При тестировании высоконагруженных приложений 58% тестируемых достигали наибольшей скорости работы в MFI (причём у половины тестируемых отрыв MFI от остальных интерфейсов был существенным). Более быстрая работа с верхней панелью отмечена всего лишь для 32% тестируемых, и только 10% проявили наибольшую эффективность с боковой панелью.

В целом, интерфейс, наиболее удобный для пользователя, предсказуемо вносил наименьший вклад в физическую нагрузку и эмоциональное состояние. Более интересным оказалось распределение интерфейсов, оказывающих на пользователя наибольшую нагрузку (в особенности эмоциональную, с учетом того, что едва ли речь идёт о положительных эмоциях). Среди пользователей, для которых наиболее оптимальна верхняя панель, самый большой вклад в ЧСС вносила боковая панель, а наибольший эмоциональный вклад распределялся между MFI и боковой панелью в соотношении 2:3. Среди пользователей, для которых оптимален MFI, максимальный вклад ЧСС распределился практически поровну между боковой и верхней панелью, а наибольший эмоциональный вклад опять же вносила боковая панель.

При тестировании графических редакторов в роли средненагруженных интерфейсов наблюдалась противоположная картина: более 70% пользователей устойчиво показали худший результат при использовании MFI. Это может оказаться следствием выбора целевых приложений. Согласно закону Фиттса, перемещение мыши бывает либо точным, либо быстрым, и переключение с одного режима на другой отнимает у оператора время. Нами отмечалось ранее [4], что большинство пользователей при горизонтальном перемещении мыши тяготеет к точному режиму больше, чем при вертикальном (в сущности, рабочее пространство анизотропно). Прецизионные движения более востребованы в графических редакторах, и это вносит вклад в улучшение результата боковой панели. Однако подобрать другие средненагруженные приложения с MFI оказалось нетривиально: во многих MFI-приложениях (обозреватель файлов Windows 8, Foxit PDF Reader и др.), панели MFI слабо задействованы в рабочем процессе и очевидно, носят скорее декоративную роль.

Ситуация с ленточными интерфейсами в GNU/Linux

Таким образом, использование MFI-подобных вариантов панелей в офисных пакетах (и, вероятно, некоторых других приложениях со сложным интерфейсом) под Linux можно признать как минимум небессмысленной альтернативой (по мнению авторов, результаты тестов свидетельствуют не столько о достоинствах MFI, сколько о недостатках традиционных панелей с большим количеством виджетов; однако сути это не меняет).

На текущий момент основной источник ленточных приложений в GNU/Linux — портированные коммерческие продукты. Если говорить о freeware-приложениях, то это WPS Office, перенесенный с платформы Android в виде кросс-платформенной настольной версии с интерфейсом, опционально копирующим MS Office 2010. WPS для Linux имеет статус беты и изобилует типичными недостатками проприетарных приложений: пытается подменять системную базу mime-типов при каждом запуске, зависит от конкретной минорной версии Qt, проявляет латентность GTK-версии, содержит инструкции по установке исключительно на китайском языке и т. д. Как пример больших платных приложений можно отметить Matlab (начиная с версии 2012).

Можно найти следы интереса к ленточным интерфейсам и в свободных проектах. Два активных обсуждения MFI разработчиками

LibreOffice вылились в появление рабочих эскизов и действующего прототипа NotebookBar (от которого было решено отказаться по прошествии нескольких месяцев). Условно можно отнести к MFI интерфейс свободного онлайнового офисного пакета ONLYOFFICE, использующего многострочную панель инструментов, визуально схожую с интерфейсом MS Office, но без вкладок (в силу более скромного набора возможностей). Благодаря недавнему переходу с ASP.NET на mono, ONLYOFFICE на текущий момент — единственное известное авторам свободное Linux-приложение с MFI-подобной разметкой.

Проблемы проектирования приложений

Помимо сомнений в эргономике, проблемы проектирования MFI-подобных приложений в GNU/Linux сводятся либо к технической трудоёмкости, либо к юридической неоднозначности.

На начальном этапе основной проблемой создания MFI-приложений для Windows было отсутствие фреймворка, избавляющего разработчиков от ручного подбора всех виджетов и программирования логики их сокрытия/показа в панелях. Конечно же для разработки ленточных интерфейсов под GNU/Linux эта проблема актуальна.

Кроме того, изначально Майкрософт предлагала лицензии на использование ленточной модели. В связи с этим в рассылке LibreOffice обсуждалась разработка MFI-варианта как загружаемого расширения — для защиты основного продукта от возможных патентных претензий. Отметим, что в ряде публикаций обосновывается невозможность успешной защиты MFI-патентов из-за наличия предшествующих аналогов (чаще всего упоминаются инструментальные панели с вкладками в RAD-системах Borland и Lotus eSuite). Подтверждением сомнительности патентной защиты MFI можно считать также исчезновение предложений о лицензировании с сайта Майкрософт.

Литература

[1] *Гоманова Е., Костюк Д.* Применение модели периферического зрения в GUI // Сетевые решения, №6, 2007. – С. 34–39.

[2] *Костюк Д. А., Латий О. О.* Модуль инструментальной оценки состояния пользователя // Открытые технологии: сб-к материалов одиннадцатой международной конференции разработчиков и пользователей

свободного программного обеспечения Linux Vacation / Eastern Europe 2015, Гродно, 25–28 июня 2015 г. – Брест, Альтернатива, 2015. – С. 91–95.

[3] *Костюк Д. А., Латий О. О., Маркина А. А.* Оценка эффективности мультипрограммной работы оператора в современном графическом интерфейсе // Десятая конференция разработчиков свободных программ: Тезисы докладов. – Калуга, 20–22 сентября 2013 года. М.: Альт Линукс, 2013. – С. 51–56.

[4] *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.

Виталий Зарубин, Иван Хахаев, Евгений Шаповалов
Санкт Петербург, Университет ИТМО, АО «НИИ ПС»

Возможность сравнительной оценки электронных образовательных ресурсов

Аннотация

В докладе ставится задача сравнения разнородных электронных образовательных ресурсов (ЭОР) с целью выбора оптимального решения для поддержки учебного процесса. Описываются свойства, которые могут быть выделены при сравнении, также предлагается методика экспертной оценки указанных свойств. Приводятся предварительные результаты сравнения распространённых ЭОР.

Электронные образовательные ресурсы (ЭОР), включая платформы для создания систем дистанционного обучения, готовые сервисы (сайты) и мультимедийные комплексы, являются существенной составной частью системы современного образования. Поэтому задача выбора «лучшего» варианта решения для поддержки учебного процесса является актуальной.

Пользователи ЭОР (включая авторов контента) в общем случае не являются специалистами в области информационных технологий, поэтому при решении задачи выбора ЭОР наряду с технологическими характеристиками должны учитываться и потребительские аспекты.

Анализ материалов по оцениванию качества ЭОР выявил следующее:

- комплексный подход к оценке и сравнению ЭОР с позиций пользователя (потребителя контента или разработчика курсов для электронного обучения) практически отсутствует (не применяется);
- многие из характеристик ЭОР являются качественными (не количественными) и допускают только субъективную оценку;
- опубликованные подходы к оценке ЭОР не учитывают их технологические характеристики.

Попытка использования ГОСТ 28195-89 «Оценка качества программных средств. Общие положения» показывает, что приведённая в данном стандарте номенклатура показателей качества не распространяется на ЭОР, которые попадают в группу 509 «прочие программные средства».

С учётом результатов анализа существующих вариантов ЭОР и имеющихся материалов по обсуждению качества ЭОР предлагается набор из шести основных свойств, существенных для сравнительной оценки ЭОР пользователем, к которым относятся:

- адаптивность;
- взаимодействие с пользователем;
- содержательность ресурса (контент);
- управление/администрирование;
- функциональная расширяемость;
- ценовая политика.

Смысловое наполнение таких названий свойств ЭОР раскрывается в докладе. Для указанных свойств можно получить качественные оценки. С точки зрения поставленной задачи эти оценки должны отражать предпочтения пользователя. Поэтому предлагается применить пятибалльную шкалу: от 1 — самой низкой оценки свойства, до 5 — наивысшего уровня в конкретном случае. Оценка 0 («ноль») означает отсутствие какого-либо свойства у оцениваемого ЭОР.

Получение оценок указанных свойств ЭОР предполагает проведение процедуры экспертного опроса и обработку его результатов. При этом получается средняя оценка для каждого свойства ЭОР, а также определяется степень согласованности экспертов в оценивании ЭОР.

В данном случае важен выбор экспертов из профессионального сообщества, поскольку оценке подвергаются не только потребительские, но и технологические качества, которые опосредованно влияют на потребительские качества.

Соответственно, согласованность мнений экспертов оценивается по разбросу оценок вокруг среднего значения (дисперсий) с применением критерия Фишера.

Предложенный подход позволяет не только получить количественную оценку разнородных ЭОР, но и сравнивать их между собой по различным показателям, что позволит пользователю сделать обоснованный выбор. Такой подход охватывает существующие виды ЭОР и оставляет возможность оценивания вновь появляющихся реализаций ЭОР. В докладе приводятся примеры оценок свойств некоторых распространённых ЭОР, проведённых по обсуждаемой методике.

Антон Уймин, Арсений Мешков
Екатеринбург, Уральский радиотехнический колледж им. А. С. Попова

СПО в СПО

Аннотация

В данной статье рассматривается опыт внедрения свободного программного обеспечения в среднем профессиональном образовательном учреждении — колледже. Приведён перечень специальностей и специализированных лабораторий, в которых применяется СПО. Описаны преимущества и недостатки использования СПО в учебном процессе и для его обслуживания.

«СПО для нищебродов!» — с этой фразы одного из представителей сообщества СПО в Екатеринбурге хотелось бы начать данную статью. Уральский радиотехнический колледж им. А.С. Попова находится в Екатеринбурге. У нас реализуется всего 12 специальностей. Из них 6 по IT-направлению. До 2008 года в нашем учебном заведении из СПО были шлюз и сайт на FreeBSD. В учебном процессе СПО не использовалось никак. В 2008 году мы начали сотрудничать с отделом «К», консультировались у них, отправляли на практику студентов. В этом же году начался перевод учебных материалов на Linux.

В колледже 19 компьютерных лабораторий. Парк ПК насчитывает около 500 шт. Парк серверов 20 шт. Обслуживанием лабораторий

занимаются зав. лаб, которые являются и преподавателями спец. дисциплин [1].

Первой лабораторией, переведенной на Linux, стала лаборатория 104 «Программно-аппаратной защиты объектов сетевой инфраструктуры». Выбор дистрибутива, который будет штатным в лаборатории был долгим и сложным. Мы прошли путь от FreeBSD → PC-BSD → Debian → Fedora → ALT Linux. У ALT Linux подкупила простота начального вхождения, качество документирования и форум, на котором действительно помогают. Мы начали с ALT Linux Школьный, сейчас работаем на ALT Linux Centaurus P7 и с нетерпением ждем выхода 8 платформы. В таблице 1 приведён перечень лабораторий и указан год перехода на СПО. Во всех лабораториях по 12 рабочих мест, кроме 304, где 27 раб. мест.

Таблица 1: Перечень лабораторий

№	**Название лаборатории**	**Год перехода**	**Преподаются дисциплины связанные с**	**На рабочем месте студента**
101	Сборки, монтажа и эксплуатации средств вычислительной техники.	2012	аппаратным обеспечением ПЭВМ. системными службами и сервисами.	ПЭВМ, принтер, сканер.
104	Программно-аппаратной защиты объектов сетевой инфраструктуры	2009	сетевыми операционными системами, информационной безопасностью.	2 ПЭВМ
114	Сетевая академия Cisco.	2013	сетевыми технологиями в рамках Cisco CCNA.	ПЭВМ
117	Лаборатория беспроводных технологий	2014	сетевыми технологиями: беспроводные сети.	ПЭВМ, с дискретными беспроводными и проводными адаптерами. В стенд входят 2 Wi-Fi маршрутизатора и IP-камера.
304	Информатики. Программирования и баз данных.	2015	информатикой и программированием.	ПЭВМ
305	Программного обеспечения компьютерных сетей. Авторизованный центр Cisco Academy	2015	основами компьютерной грамотности.	ПЭВМ

Есть участки на которых нет возможности перейти на СПО и приходится использовать Windows. В связи с тем, что мы обучаем для

конкретных организаций, то прикладное ПО диктуют они. Используются такие ПП как Altium Designer, Autocad, MS Office, 1С и т.д.

Стандартным пакетом дополнительного ПО в лабораториях является: Remmina, Firefox, Geany, iTALC, OpenSSH, PuTTY, VirtualBox.

К нестандартному относятся такие пакеты как Cisco Packet Tracer, LinSSID, Aircrack-ng, Wireshark, Metasploit Framework.

Сервисы на базе СПО, используемые для обслуживания образовательного процесса, приведены в таблице 2.

Таблица 2: Используемые сервисы

Сервис	**На чем настроен**	**Назначение**
Система видеонаблюдения	Debian 7 + MotionEye	Обеспечение безопасности лабораторий, отслеживание инцидентов
Зеркало репозиториев Debian	Debian 7 + apt-mirror	Проведение лабораторных работ
Зеркало репозиториев ALT Linux	ALT Linux P7 + sisyphus-mirror	Обслуживание ПЭВМ
Сервер сетевой загрузки	Debian 7 + tftpd-hpa + isc-dhcp-server + syslinux + nfs-kernel-server + smb + apache2	Обслуживание ПЭВМ
Вики-энциклопедия	ALT Linux P7 + MediaWiki	Создание и хранение документации
Сервер SNMP-мониторинга	Debian 8 + Zabbix	Мониторинг сетевого оборудования
Система дистанционного обучения	FreeBSD 7 + Moodle	СДО

Сервер сетевой загрузки позволяет производить загрузку следующих образов ОС — Debian 8, ALT Linux P7, Windows 7/8.1/2012R2, а так же служебных утилит HDT, memtest86+, MS DaRT, MHDD.

В 2016 году планируется развёртывание LDAP-домена на базе ALT Linux, так же начали сотрудничество с компанией РусБИТех, запланирована организация учебных мест на базе Astra Linux SE.

Вместо вывода:

1. Свободное программное обеспечение позволяет айтишникам более глубоко изучать технологии: если студент смог настроить DNS на Linux, то у него не возникнет проблем при аналогичных настройках на Windows или сетевом оборудовании.

2. Кризис и санкции стимулировали внедрение СПО в СПО. Например, до 2015 года руководство особого интереса не проявляло, а в 2015 году мы выступали перед советом директоров

ССУЗов СО с докладом о практике внедрения свободного ПО в нашем колледже.

3. Свободное программное обеспечение сложнее в развёртывании, настройке и обслуживании. Руководство не готово оплачивать ни полноценную техническую поддержку, ни доплачивать за обслуживание СПО, что влияет на мотивацию технического персонала.

4. Так как у Linux отсутствует маркетинговая поддержка, то затруднительно агитировать и продвигать свободные технологии в массы. Например, колледж проводит много различных мероприятий: областные и международные олимпиады. С одной стороны, поддержку данным мероприятиям готовы оказать только люди, интересующиеся СПО в частном порядке. Компании и вендоры не готовы оказывать поддержку. С другой стороны, большое количество учебных заведений испытывают трудности при подготовке студентов к заданиям с использованием СПО.

5. К сожалению, мы нищеброды, которые пытаются готовить профессионалов, в чём СПО нам активно помогает.

Литература

[1] *Уймин А. Г., Ершова К. О.*, Перспективы развития специальностей информационно-технического профиля в рамках СПО [Текст] // Теория и практика науки третьего тысячелетия: сборник статей Международной научно практической конференции. 7 марта 2014 г. / Под общ. ред. А. А. Суакисян. — Уфа: РИЦ БашГУ, 2014. — С. 275–279 — ISBN 978-5-7477-3501-9

Амелькин Сергей Анатольевич, Топоркова Юлия Аркадьевна
Переславль-Залесский, Частное образовательное учреждение высшего образования Институт программных систем «УГП имени А.К.Айламазяна»

Алгоритмическое обеспечение для задачи мониторинга качества обучения

Аннотация

Качество обучения — термин, интуитивно понятный, но неоднозначно интерпретируемый. Качество обучения может быть оценено экспертом, как степень соответствия реального образовательного процесса и некоторой эталонной модели, причем модель эта может быть различной у разных экспертов. В образовательной системе с большим количеством экспертов, выступающих в различных ролях, эталонные модели не могут быть сведены к некоторой объективной эталонной модели, относительно которой можно рассчитывать показатель (индекс) качества. Тем не менее именно такой подход наиболее распространен для оценки эффективности работы учебных учреждений, что приводит к подмене целей и конкурентной борьбе в образовательных системах. Вместе с тем, для описания неформализуемых показателей существуют алгоритмы коллаборативной фильтрации, которые позволяют обеспечить по экспертным оценкам области «максимумов» этих показателей с учетом различий предпочтений экспертов. В докладе представлены алгоритмы, реализуемые в свободно распространяемых программных продуктах, которые могут быть адаптированы для задач оценки качества обучения.

Социальные и экономические системы характеризуются тем, что поведение подсистем определяются не только количественными параметрами, экстенсивными или интенсивными, но также и различием критериев, что связано с различными ролями подсистем. Описание ролей можно осуществить путем введения неформальных и неформализуемых параметров и критериев. Неформальными называем параметры, которые могут быть неоднозначно интерпретируемы экспертами, неформализуемыми — параметры, которые не могут быть выражены числовым образом. Одним из таких критериев в социальных системах является качество образования.

Важность задач мониторинга и управления качеством образования приводит к введению различных моделей, сводящихся к построению скалярных или векторных индексов качества. Индексы строятся,

как взвешенная сумма наблюдаемых или оцениваемых параметров, которые, по мнению экспертов, разрабатывающих эти модели, коррелируют с качеством образования. Наблюдаемые параметры могут быть выражены числовыми величинами, оцениваемые — могут быть представлены, как балльная оценка. Предельное значение индекса соответствует эталонной модели системы. Однако, использование индексов, основанных на экспертных оценках, приводит к следующим негативным эффектам:

- Различные эксперты руководствуются при выставлении оценок различными эталонными моделями, что приводит к разным оценкам одного и того же образовательного учреждения, то есть оценка качества обучения зависит от личности эксперта;
- Так как сами образовательные учреждения могут выполнять различные роли, то их критерии качества могут различаться, что не учитывается при построении индекса;
- Если известен алгоритм построения индекса, то образовательное учреждение может быть вынуждено работать в условиях подмены целей: вместо повышения качества образования, учреждение стремится достичь предельных показателей, входящих в индекс;
- Стремление достичь соответствия с эталонными моделями приводит к подмене роли образовательного учреждения в системе;
- Если значение индекса качества или место образовательного учреждения в рейтинге, построенном на основе такого индекса связано с финансовыми или организационными преференциями, то использование индекса приводит к конкурентной борьбе между образовательными учреждениями, что негативно влияет на общую эффективность социальной системы.

Тем не менее, при понимании всех недостатков индексов качества образования, именно эта модель наиболее распространена, так как наиболее понятна и не требует специальных математических знаний для анализа состояния социальной системы. При этом сам анализ социальной системы может иметь различные цели: от мониторинга, предполагающего самоорганизацию социальной системы, до управления, предполагающего стандартизацию процесса обучения так, что следование стандартам обеспечит соответствие эталонной модели.

Задача построения алгоритмического и программного обеспечения для оценки качества обучения, при этом свободного от указанных

недостатков, может быть сформулирована, как: определить направления развития образовательного учреждения, которые могут привести к повышению качества образовательного учреждения в соответствии с его ролью в социальной системе. Такая постановка задачи совпадает с задачей поиска эталонных моделей и наиболее близких к ней образовательных учреждений. Тем не менее, требование ролевой обусловленности не позволяет сформировать оптимальный в некотором смысле индекс. Можно предложить другие методы решения поставленной задачи.

Если ставится задача мониторинга состояния социальной системы, то можно построить фазовый портрет наблюдаемых и оцениваемых показателей образовательных учреждений, включающий как текущие показатели, так и скорости их изменения. Так как мониторинг предполагает самоорганизацию социальной системы, то необходим кластерный анализ направлений самоорганизации, который может быть проведен с использованием метрики Евклида-Махаланобиса. Общая архитектура программного комплекса, реализующая решение задачи кластеризации образовательных учреждений по наблюдаемым и оцениваемым параметрам, включает решение оптимизационной задачи определения центров кластеров, минимизирующее среднее расстояние до состояний образовательных учреждений в течение заданного периода времени при сохранении тенденции развития. Производя анализ результатов экспертного анализа, такой программный комплекс позволит выявить роли образовательных учреждений и оценить направленность их развития. Кроме того, он обеспечивает визуализацию ситуации, в том числе в когнитивном виде.

Для решения задач кластеризации необходимо найти матрицы ковариаций всех классов на основе известных выборок. Затем с помощью подсчета расстояний от заданной точки до каждого кластера выбрать кластер, для которого расстояние минимально. При этом предполагаем, что кластеры в фазовом пространстве разделены гиперплоскостями. Для решения такой задачи требуется построить дискриминантную функцию, позволяющую для любых двух кластеров определить принадлежность заданной точки к одному из них. Для решения задач кластеризации требуется на полученных данных (обучающая выборка) сформировать кластеры, соответствующие различным ролям, а затем для каждого нового исследования можно поставить в соответствие роль и направление развития.

Объекты	**Признаки и их значения**			**Класс**
	x_1	...	x_p	
ω_1	x_{11}	...	x_{1p}	Ω_1
...	...	...	...	...
ω_{m_1}	$x_{m_1 1}$	...	$x_{m_1 p}$	Ω_1
ω_{m_1+1}	$x_{(m_1+1)1}$	...	$x_{(m_1+1)p}$	Ω_2
...	...	...	...	...
ω_m	x_{m1}	...	x_{mp}	Ω_2

Таблица 1: Обучающая выборка классов Ω_1 и Ω_2

Пусть задана исходная информация для объектов $\omega_i, i = 1, \ldots, m$ в виде обучающих выборок (таблица 1). Классы Ω_1 и Ω_2 представлены матрицами значений признаков X_1 и X_2 размерности $(m_1 \times p)$ и $(m_2 \times p)$ соответственно, причем $m_1 + m_2 = m$.

Будем решать задачу определения расстояния между классами, а также задачу определения расстояния между точкой (как частным случаем некоторого неизвестного класса) и классами Ω_1 и Ω_2.

Одним из возможных подходов к решению задачи классификации служит метод построения дискриминантной функции. Дискриминантная функция имеет вид $f(x) = AX$, где $A^T = (a_1, a_2, \ldots, a_p)$ — коэффициенты гиперплоскости, разделяющей классы (коэффициенты дискриминантной функции); $X = (x_1, x_2, \ldots, x_p)$ — вектор признаков, определяющих точку, через которую проходит гиперплоскость.

Решение задачи классификации сводится к построению на базе дискриминантной функции распознающей функции вида: $F(x) = f(x) - C$. Здесь $C = \frac{1}{2}\left(\bar{f}_1(x) - \bar{f}_2(x)\right)$, $\bar{f}_1(x), \bar{f}_2(x)$ — значения дискриминантных функций, проходящих через центры классов Ω_1 и Ω_2. Для изменения расстояний строится объединенная ковариационная матрица

$$S_0 = \frac{1}{m_1 + m_2 - 2}(S_1 + S_2) \tag{9.1}$$

где S_1 и S_2 — матрицы ковариаций для классов Ω_1 и Ω_2 соответственно.

Оценки внутригрупповой D_1 и межгрупповой D_2 вариаций переменных поставить и решить оптимизационную задачу $\frac{D_1}{D_2} \to \max$, решение которой и есть искомый вектор A.

Классификация тестируемого вектора ω осуществляется далее стандартно по правилу:

$$\omega \in \begin{cases} \Omega_1, & \text{если } F(x) \geq 0, \\ \Omega_2, & \text{если } F(x) < 0. \end{cases}$$

Таким образом, метрикой для случая бинарной классификации служит функция $F(x) = f(x) - C$. Особенность решения задачи — отказ от частных ковариационных матриц в пользу обобщенной матрицы. Принцип объединения ковариационных матриц может быть использован для построения обобщенной метрики.

Для задач управления целесообразно воспользоваться алгоритмами коллаборативной фильтрации, позволяющими не только решить задачу поиска объектов (в нашем случае — образовательных учреждений), наиболее точно совпадающих с эталонной моделью, но и провести их ранжирование с учетом различий эталонных моделей экспертов. Более того, используя алгоритмы коллаборативной фильтрации, можно также решить задачу кластеризации, выделив объективные эталонные объекты различных групп экспертов.

Использование методов кластеризации и коллаборативной фильтрации, можно создать программный комплекс для оценки качества обучения, не производящий подмены целей и ролей в социальной системе, что позволяет перейти от формальной процедуры к мониторингу и управлению реальными достижениями образовательных учреждений.

Литература

[1] *Амелькин С. А., Захаров А. В., Хачумов В. М.* Обобщенное расстояние Евклида-Махаланобиса и его свойства. — Информационные технологии и вычислительные системы, № 4, 2006, с. 40–44.

[2] *Сошникова Л. А., Тамашевич В. Н., Усбе Г., Шефер М.* Многомерный статистический анализ в экономике: Учебное пособие для вузов. Под ред. В.Н. Тамашевича. — М.: БНИТИ — Дана. — 1999. — 598 с.

[3] *Понизовкин Д. М., Амелькин С. А.* Оптимальное распределение проектов при проведении экспертизы // Электронные библиотеки: Перспек-

тивные Методы и Технологии, Электронные коллекции. — 2010. — с. 524–525.

[4] *Понизовкин Д. М., Амелькин С. А.* Построение оптимального графа связей в системах коллаборативной фильтрации // Программные системы: теория и приложения. 2011. — Т. 2. — № 4. с. 107–114

[5] *Понизовкин Д. М., Амелькин С. А.* Математическая модель коллаборативных процессов принятия решений // Программные системы: теория и приложения. 2011. — Т. 2. — № 4. с. 95–99.

[6] *Амелькин С. А., Понизовкин Д. М.* Оптимальное проведение экспертизы образовательных процессов // Труды XVII Всероссийской научно-методической конференции Телематика'2010, СанктПетербург: Университетские телекоммуникации. — 2010. — № 1, с. 158–159.

Пустыгин А. Н., Ковалевский А. А., Ошнуров Н. А.
Челябинск, Челябинский Государственный Университет

Построение эквивалентного представления исходных текстов программ в форме пригодной для выполнения анализов потока данных в потоке управления

Аннотация

Предложено эквивалентное представление исходного текста в форме, пригодной для выполнения анализов потока данных в потоке управления. Оно основано на трансформации универсального промежуточного представления по данным и операциям над ними. Универсальное промежуточное представление является текстовым эквивалентом абстрактного синтаксического дерева на языке разметки. Также предложен вариант анализа, который представляет собой граф распространения данных в потоке управления и позволяет отследить распространение данных из заданного носителя в потоке управления метода.

В процессе статического анализа программных систем, написанных на нескольких языках программирования, зачастую возникают задачи различного рода и уровней сложности, которые необходимо решать для каждого из используемых в системе языков. Одним из подходов, упрощающих статический анализ и построение различных эквивалентных представлений исходных текстов на разных языках,

является использование универсального промежуточного представления (УПП) [1,2]. Предложено представление, являющееся трансформацией УПП по данным и операциям над ними, которое можно использовать для построения анализов распространения и использования данных. Предложен один из таких анализов, который представляет собой граф распространения данных в потоке управления и позволяет отследить распространение данных из заданного носителя в потоке управления метода.

Представления генерируются в формате XML, который был выбран вследствие его большой распространённости и большого количества инструментов для его обработки, что упрощает построение автоматизированных инструментов анализа.

Трансформация универсального промежуточного представления по данным и операциям над ними

В дальнейшем для обозначения трансформации УПП по данным и операциям над ними будем использовать термин Universal Intermediate Representation of Data/Control Flow (UIRDCF).

Известны аналогичные представления: Program Dependence Graph (PDG) [3] и System Dependence Graph (SDG) [3,4], в которых фигурируют как зависимости данных, так и зависимости потока управления. Связи в этих представлениях, которые относятся к зависимостям данных (data dependence), строятся по факту наличия операций над носителями данных (переменными), которые упоминаются в каждом из узлов, между которыми эта связь устанавливается, а сами узлы являются единичным оператором выполнения (single execution statement). Направление связи определяется потоком управления и AST. При этом сами связи не атрибутируются информацией о типе производимой операции, вместо этого в них содержится узел AST этой операции.

Алгоритм получения представления UIRDCF [5] однопроходный и выражается в трансформации имеющейся информации в УПП в абстрагированный набор операций над идентификаторами (носителями данных) в обертке операторов потока управления (Таблица 1).

Схему данных формата UIRDCF можно найти в [6], а текстовое описание элементов схемы в [7].

Исходный текст	**Представление UIRDCF**

`if(y<x)`
`{`
`  Swap(x,y);`
`  return x;`
`}`
`else y = x;`
`return y`
	`<If>`
	`<Condition>`
	`  <Read name="y" id="2"/>`
	`  <Read name="x" id="1"/>`
	`</Condition>`
	`<Then>`
	`<Block>`
	` <Call name="Swap" id="40fff10">`
	`  <Args>`
	`  <Share name="x" id="1"/>`
	`  <Share name="y" id="2"/>`
	`  </Args>`
	`  </Call>`
	`  <Return><Copy name="x" id="1"/></Return>`
	`</Block>`
	`</Then>`
	`<Else>`
	` <Modify name="y" id="2">`
	`  <Copy name="x" id="1"/>`
	` </Modify>`
	`</Else>`
	`</If>`
	`<Return><Copy name="y" id="2"/></Return>`

Таблица 1: Пример представления UIRDCF

Граф распространения данных в потоке управления

На основе полученного эквивалентного представления был разработан анализ в форме графа распространения данных в потоке управления. Данное представление является демонстрацией возможности выполнения анализов по разработанному эквивалентному представлению UIRDCF и позволяет отследить распространение данных из заданного носителя в потоке управления метода, а также вызовы, которые могут модифицировать аргументы, с учетом передачи переменной по ссылке или через указатель. Возможны и другие анализы по данному представлению.

В дальнейшем граф распространения данных в потоке управления будем называть Data Redistribution in Control Flow (DRCF). Данное представление является трансформацией UIRDCF по заданным параметрам:

1. Отслеживаемый идентификатор (носитель данных);
2. Точка начала отслеживания (позиция в исходном тексте);
3. Глубина отслеживания (не ограничена по умолчанию);
4. Список имен методов с номерами и уровнями аргументов для отслеживания.

Алгоритм построения однопроходный. Построение начинается с обнаружения, начиная с заданной, начальной точки (позиции в исходном тексте) отслеживаемого идентификатора, который добавляется в *список отслеживаемых идентификаторов*. Далее стартует процесс отслеживания (трассировки) операций над идентификаторами из данного списка: чтения (Read), копирования (Copy), распространения (Share) и модификации (Modify). В процессе обработки операций копирования и распространения, идентификатор, выступающий в роли акцептора данных, также добавляется в *список отслеживаемых идентификаторов* и помечается *индексом глубины* на единицу большим своего донора, но только в случае, если он уже не отслеживается и не обладает меньшим индексом глубины. Таким образом, формируется дерево узлов модификаций и распространения данных из заданного носителя в пределах одного метода.

Для детального анализа, дополнительно указывается список методов с набором интересуемых аргументов в качестве отслеживаемых носителей (Таблица 2).

Данное представление также записывается как текст на языке разметки XML.

Сравнение предложенных эквивалентных представлений и анализа с аналогами

UIRDCF позволяет получить расширенную, но несколько абстрагированную от AST, версию PDG и SDG. Различные операторы из AST в UIRDCF представлены классом производимой операции над данными.

DRCF при этом можно сравнить с графом зависимостей данных из PDG, построенным для одной переменной, но с ответвлениями для

	Исходный текст	Представление DRCF
1		<DRCF>
2		<Trace type="method-decl" level="0" id="100" name="Swap" line="3">
3	void Swap(int &x, int &y)	<Trace type="arg" level="1" id="101" name="x" index="1" line="3"/>
4	{	<Trace type="decl" level="2" id="102" name="t" index="1" line="5">
5	int t=x;	<Trace type="copy" level="1" id="101" name="x" index="2" line="5"/>
6	x=y;	</Trace>
7	y=t;	<Trace type="modify" level="1" id="101" name="x" index="3" line="6"/>
8	}	<Trace type="modify" level="3" id="103" name="y" index="1" line="7">
9		<Trace type="copy" level="2" id="102" name="t" index="2" line="7"/>
10	int main()	</Trace>
11	{	</Trace>
12	int x = rand();	<Trace type="decl" level="0" id="1" name="x" index="1" line="12"/>
13	int y = rand();	<Trace type="read" level="0" id="1" name="x" index="2" line="14"/>
14	if(y<x)	<Trace type="share" level="0" id="1" name="x" index="3" line="16"/>
15	{	<Trace type="copy" level="0" id="1" name="x" index="4" line="17"/>
16	Swap(x,y);	<Trace type="modify" level="1" id="2" name="y" index="1" line="19">
17	return x;	<Trace type="copy" level="0" id="1" name="x" index="5" line="19"/>
18	}	</Trace>
19	else y = x;	<Trace type="copy" level="1" id="2" name="y" index="2" line="20"/>
20	return y;	</DRCF>
21	}	

Таблица 2: Отслеживается переменная «x» и 1-ый аргумент метода «Swap»

побочных носителей (Таблица 2). Побочным носителем в данном примере выступает переменная «y» (помечена жирным), которая принимает данные из отслеживаемой переменной «x» посредством операции класса копирования (в данном случае оператор присваивания).

Возможные варианты использования предложенных эквивалентных представлений

Возможные варианты использования UIRDCF:

- Фильтрация в исходном коде позиций использования заданного поля или глобальной переменной по типу операции (Read, Copy, Share, Modify).
- ЭП Butterfly — глобальные потоки данных к переменной и из неё с разграничением типов потоков (copy/share).

Возможные варианты использования DRCF:

- Точки потенциальной неявной модификации данных — передача аргументов в вызовы методов по ссылке или указателю, модификация вторичных носителей данных (акцепторов), полученных через операцию Share (через указатель или по ссылке).

Выводы

Полученное эквивалентное представление UIRDCF удовлетворяет поставленным критериям для эквивалентного представления, на основании которого можно выполнять анализы потоков данных и управления совместно. Использование в качестве его основы УПП позволяет получить ряд универсальных инструментов для построения анализов.

Представление DRCF демонстрирует возможность анализов по предложенному эквивалентному представлению UIRDCF.

Описанный анализ DRCF демонстрирует преимущество уровневого подхода к построению полезных анализов исходных текстов, путем разбиения этой задачи на универсальные и независимые фрагменты и уровни: генератор DRCF не включает в себя ни модуля анализа исходного текста, ни модуля разбора УПП, только модуль разбора представления UIRDCF; при том какие-либо изменения в формате УПП, которые соответствующим образом учтены в генераторе UIRDCF, не повлияют на генератор анализа DRCF.

Литература

[1] *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.

[2] UIR.pdf [Электронный ресурс]. URL: `https://yadi.sk/i/u3v0H-QTeohLV` (дата обращения: 08.01.2016).

[3] Background. Program Dependence Graph. [Электронный ресурс]. URL: `http://www4.comp.polyu.edu.hk/~cscllo/teaching/SDGAPI/background.htm` (дата обращения: 08.01.2016).

[4] University of Leicester CO7206 [Электронный ресурс]. URL: `http://www.cs.le.ac.uk/people/mer14/quiz2answer.html` (дата обращения: 08.01.2016).

[5] *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.

[6] dcflow.xsd [Электронный ресурс]. URL: `https://yadi.sk/d/E0DfDdwKdLvjF` (дата обращения: 08.01.2016).

[7] UIRDCF.txt [Электронный ресурс]. URL: `https://yadi.sk/d/yWhYELCkeoiYS` (дата обращения: 08.01.2016).

Пустыгин А. Н., Ковалевский А. А., Белоусов И. С.
Челябинск, Челябинский Государственный Университет

Построение эквивалентного представления зависимостей классов, полей, методов, функций и их перекрестного использования в исходных текстах программ

Аннотация

Предложено эквивалентное представление исходного текста в форме графа зависимостей классов, полей, методов, функций и их перекрестного использования. Оно основано на трансформации универсального промежуточного представления в текстовый формат на языке описания графов. Универсальное промежуточное представление является текстовым эквивалентом абстрактного синтаксического дерева на языке разметки.

Для упрощения статического анализа многоязычных программных систем, включающего построение эквивалентных представлений и выполнения различного рода анализов, предлагается использование универсального промежуточного представления (УПП) [1,2]. Предложено представление, являющееся трансформацией УПП, в форме графа зависимостей классов, полей, методов, функций и их перекрестного использования. Предложен прототип генератора такого представления в текстовый формат на языке описания графов DOT.

Граф зависимостей классов, полей, методов, функций и их перекрестного использования

Для построения графа зависимостей классов, полей, методов, функций и их перекрестного использования (в дальнейшем — «граф зависимостей») путем трансформации УПП необходим полный доступ к DOM УПП, которое в свою очередь представлено XML документом. В качестве средства загрузки и обработки DOM УПП была выбрана библиотека pugixml [3]. Критерием выбора послужил сравнительный тест [4].

Так как предложенное эквивалентное представление предлагается в качестве конечного узла перед непосредственным выполнением пользовательского анализа, необходимо выбрать было формат визуализации. Из существующих форматов представления графов был выбран язык описания графов DOT, как один из самых простых и распространенных, а также пакет утилит Graphviz для автоматической визуализации графов в формате DOT.

Алгоритм генерации графа зависимостей двухпроходный. Дерево УПП имеет в контексте графа зависимостей структуру как на рисунке 1. Загруженная DOM рекурсивно обходится в глубину и информация о каждом узле из контекста графа зависимостей сохраняется в индекс — DOM2. После обхода происходит генерация графа зависимостей в файл на языке описания графов DOT путем обхода дерева DOM2 в

глубину. Граф зависимостей содержит программные сущности (классы, поля, методы и т.д.) в качестве узлов, а связи атрибутированы в соответствии с типом сущностей и их отношением. Атрибутом связей выступают: вес связи, цвет, начертание.

Связи наследования и принадлежности к классу обозначены зеленой сплошной связью.

Связи вызова метода — сплошной стрелкой со счетчиком повторов. Синие стрелки обозначают статические вызовы, красные — константные, остальные — черным.

Связи использования поля обозначены черной сплошной стрелкой со счетчиком повторов.

Вершины графа зависимостей представлены тремя типами: класс, поле, метод/функция. Их общий вид представлен на рисунках 2,3,4.

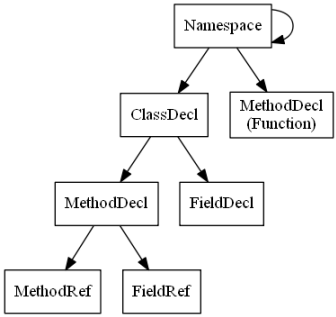


Рис. 1: Структура УПП файла в контексте графа зависимостей

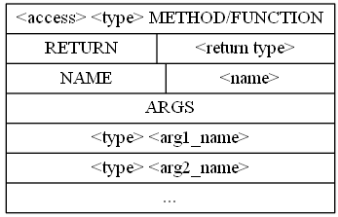


Рис. 2: Вид узла методов классов и функций

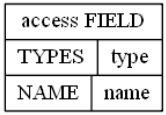


Рис. 3: Вид узла поля класса

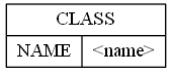


Рис. 4: Вид узла класса

В качестве тестового проекта было использованно УПП исходного текста проекта на C++ библиотеки pugixml [3,5].

Прототип генератора графа зависимостей имеет функционал получения среза по классам и методам, позволяющий визуализировать только интересуемую часть графа (Рисунок 5).

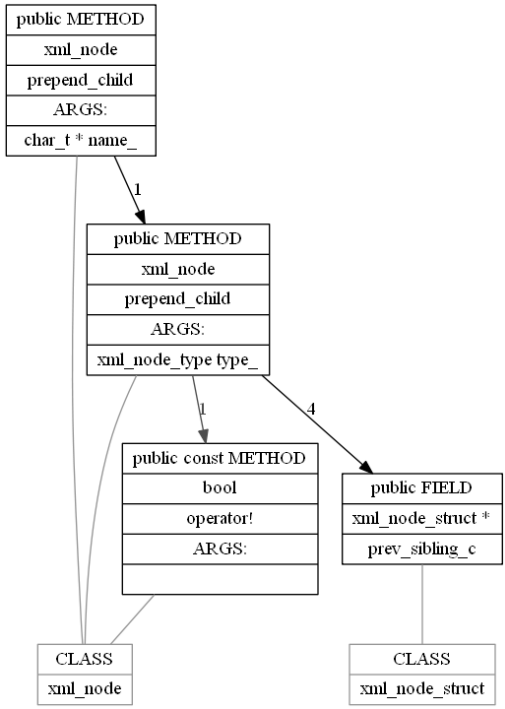


Рис. 5: Пример среза графа зависимостей

Выводы

Предложено эквивалентное представление исходного кода программ в форме графа зависимостей классов, полей, методов, функций и их перекрестного использования. Разработан прототип генератора представления в формате языка описания графов DOT. Генератор имеет функционал получения среза по классам и методам, позволяющий выполнять анализ интересуемых зависимостей.

Предложенное эквивалентное представление основывается на универсальном промежуточном представлении и не зависит от языка программирования исследуемого исходного текста, что означает его собственную универсальность и демонстрирует преимущество уровневого подхода к построению полезных анализов исходных текстов, путем разбиения этой задачи на универсальные и независимые фрагменты и уровни.

Литература

[1] *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.

[2] UIR.pdf [Электронный ресурс]. URL: `https://yadi.sk/i/u3v0H-QTeohLV` (дата обращения: 08.01.2016).

[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: 15.01.2016)

[4] Pugixml Benchmark [Электронный ресурс]. URL: `http://pugixml.org/benchmark.html` (дата обращения: 15.01.2016)

[5] pugixml_uir.xml [Электронный ресурс]. URL: `https://yadi.sk/d/gznJ7uXTmzXGJ` (дата обращения: 15.01.2016)

Евгений Алексеев, Денис Лутошкин, Наталья Болтачева
Киров, Вятский государственный университет
http://evgeniyalekseev.wordpress.com,
http://distributiv.wordpress.com

Использование свободного программного обеспечения в курсе «Операционные системы»

Аннотация

Представлен опыт использования свободного программного обеспечения в курсе «Операционные системы» на кафедре «Прикладная математика и информатика» Вятского государственного университета.

В 2014-2015 уч. году в Вятском государственном университете на кафедре «Прикладная математика и информатика» был перестроен курс «Операционные системы». Эта дисциплина была предложена студентам на базе свободных операционных систем семейства Linux. Курс «Операционные системы» студентам специальности «Прикладная математика и информатика» читается в четвертом семестре. По учебному плану предусмотрены 18 часов лекций, 18 практических занятий, 36 часов лабораторных работ и курсовой проект. По завершении изучения предмета студенты сдают экзамен.

Теоретический курс включает следующие разделы:

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

На практических занятиях и лабораторных работах студенты знакомятся с операционной системой семейства Linux и реализацией в ней основных принципов функционирования операционных систем. В 2014-15 уч. году студентам были предложены следующие темы:

- 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.

Целью курсового проекта является построение специализированного дистрибутива операционной системы. Было предложено самостоятельно собрать дистрибутив определённой направленности (офис-

ный, для программиста и т. д.). Студенты самостоятельно определяются с базовым дистрибутивом и изучают технологию сборки. Проводят отбор свободного программного обеспечения, включаемого в свой дистрибутив. Разработанные студентами дистрибутивы доступны для ознакомления всем желающим на странице `https://distributiv.wordpress.com/distributions`. Там же представлена инструкция по сборке своего дистрибутива на базе ОС Ubuntu Linux. Во время выполнения курсового проекта студенты понимают, что именно принципы свободного программного обеспечения позволяют быстро и эффективно разрабатывать специализированные операционные системы. Выполнение курсового позволяет довольно быстро освоить ОС семейства Linux на уровне продвинутого пользователя и использовать Linux в повседневной практике, в том числе и при дальнейшем изучении предметов в университете.

Михаил Ройтберг
Москва, ФГУ ФНЦ НИИСИ РАН
Проект: Информатика в школе `http://ege-go.ru`

Информатика в школе: стандарты, программы, экзамены, учебники, интернет-ресурсы

Аннотация

Преподавание в высшей школе зависит от того, чему абитуриенты научатся в школе. В докладе представлено современное состояние преподавания информатики, включая стандарты преподавания, примерные программы, ЕГЭ. Также представлены наиболее интересные (по мнению автора) учебники и Интернет-ресурсы.

В 2009-2012 гг были приняты новые Федеральные Государственные Образовательные Стандарты (ФГОС) общего образования — отдельные стандарты для начальной (1-4 классы), основной (5-9 классы) и средней (10-11 классы) школы. Эти стандарты пришли на смену ФГОС, принятым в 2004-м году. Новые стандарты носят рамочный характер — в них для каждого предмета указаны только основные планируемые результаты. Что именно должно изучаться и а каком объеме определяется отдельными документами — примерными основными образовательными программами (ПООП). ПООП по информа-

тике для основной школы была утверждена в 2015 г. Работа над ПООП для средней школы должна быть завершена в текущем году.

В соответствии с ПООП в основной школе на изучение информатики выделяется 105 академических часов (при 35 учебных неделях в учебном году), при наличии возможностей у школы это число может быть увеличено до 175 часов. В средней школе ФГОС предусматривает возможность изучения информатики на двух уровнях — базовом (70 часов) и углубленном (280 часов). Говоря неформально, обучение на базовом уровне рассчитано на учеников, которые не предполагают связать свою профессиональную работу с программированием или ИТ-технологиями. На уроках информатики они предположительно повторяют наиболее важные темы из курса основной школы и знакомятся с ИТ-технологиями. На углубленном уровне курс информатики средней школы существенно расширяет знания, полученные учениками в основной школы.

Новые стандарты вводятся в действие постепенно. Так, Основной Государственный Экзамен (ОГЭ, 9-й класс) будет приниматься в соответствии с новыми ФГОС только с 2018 г., а ЕГЭ (11-й класс), по-видимому, с 2020 г. Тем не менее, уже сейчас содержание ОГЭ и ЕГЭ учитывает ориентиры новых ФГОС, а вновь создаваемые учебники должны соответствовать новым ФГОС.

Материал, изучаемый в курсе информатики, включает 3 основных блока:

- Математические основания информатики
- Алгоритмы и программирование
- Использование программ и сервисов; ИКТ-технологии.
- Каждый из этих блоков изучается как в основной, так и в средней школе.

В блоке «Математические основания информатики изучаются следующие темы (названия условные): (1) Кодирование и передача данных; (2) Системы счисления (3) Дискретные комбинаторные объекты; (4) Элементы комбинаторики и теории множеств; (5) Элементы математической логики.

Блок «Алгоритмы и программирование» — основной блок курса, что находится в соответствии с ФГОС. Изучение этого блока, как в основной школе, так и в средней школе (при углубленном уровне изучения) предусматривает программирование в выбранной среде про-

граммирования; практическая работа должна быть основным инструментом при изучении этого блока. С точки зрения теории (ниже не уточнено, что изучается в основной, а что — в средней школе), школьники изучают основные конструкции последовательного программирования, включая рекурсию, алгоритмы обработки чисел, включая, например, алгоритм Евклида и алгоритм перевода числа из двоичной записи в десятичную, а также однопроходные алгоритмы обработки числовых последовательностей (пример: определение суммы всех нечетных элементов), алгоритмы сортировки. Школьники также знакомятся с алгоритмами, использующими идею динамического программирования (определение минимального пути в графе, подсчет количества путей). Представлены также основные характеристики сложности выполнения алгоритма — время выполнения и используемая память.

В блоке, посвященном программным ресурсам и ИКТ, рассмотрены основные виды программного обеспечения, в том числе — прикладного (текстовые процессоры, электронные таблицы, базы данных и пр.), средства цифрового представления и обработки аналоговых данных, устройство современных компьютеров, тенденции развития архитектуры компьютеров и ИКТ

Единый Государственный Экзамен за последние годы претерпел значительные изменения и в настоящее время, по-видимому, в основном, сложился. Он включает 27 заданий по всем указанным выше блокам курса, в том числе 23 задания с кратким ответом (правильность ответа проверяется формально путем сравнения с эталоном) и 4 задания с развернутым ответом (проверяется экспертной комиссией). Среди заданий 12 относятся к базовому уровню сложности, 11 — к повышенному и4 — к высокому. По тематике задания распределены следующим образом: «Математические основания информатики» — 11 заданий (5 заданий базового уровня, 4 задания повышенного уровня, 2 задания высокого уровня); «Алгоритмы и программирование»- 11 заданий (3 задания базового уровня, 6 заданий повышенного уровня, 2 задания высокого уровня); «Технологии» — 5 заданий (4 задания базового уровня, 1 задание повышенного уровня). В настоящее время ЕГЭ проводится в «бумажной технологии»; с этим связано, в частности, малое количество заданий последнего блока.

Ввиду ограничений на объем тезисов мы не приводим здесь анализа существующих учебников и интернет ресурсов.

Анатолий Кушниренко, Михаил Ройтберг, Денис Хачко, Виктор Яковлев
Москва, Пущино, ФГУ ФНЦ НИИСИ РАН

Проект: КуМир http://niisi.ru/kumir, http://gitorious.org/kumir2

Кумир 2.1: современное состояние проекта

Аннотация

КуМир (Комплект Учебных МИРов) — система программирования, предназначенная для поддержки начальных курсов информатики и программирования. В 2015 году в школах и началось использование системы Кумир 2.0. Кумир 2.1 — это продолжение развития предыдущих версий системы Кумир.

Состав Кумир 2.1

В состав системы входят три варианта графического интерфейса, компилятор языка Кумир в выполнимыи баит код, компактныи интерпретатор баит-кода, шесть графических исполнителей, модуль поддержки практикумов.

Варианты графического интерфейса:

- **Классический** — Максимально похожий на Кумир версий 1.8-1.9 привычный интерфейс, с поддержкой прикрепления дочерних окон к главному окну. Предназначен для 5-7 классов.
- **Про** — Интерфейс основанный на вкладках, что дает возможность держать открытыми сразу несколько Кумир-программ.
- **Учительский** — Аналогично варианту «Про», но есть поддержка «Учительских» функций: защиты строк от редактирования и создание проверяющих алгоритмов скрытых от учеников.

Графические исполнители входящие в систему 2.1:

- **Робот** — классический, наиболее часто используемый исполнитель.
- **Рисователь** — создает рисунки на листе, предназначен для обучения основам растровой графики (разработан совместно с К.Ю.Поляковым по его предложению).
- **Водолей** — задачи на переливание.

- **Кузнечик** — задачи на числовой прямой.
- **Чертежник** — векторная графика; в версии 2.1 в исполнителе появились "учительские" команды, необходимые для автоматической проверки заданий.
- **Черепашка** — создание на экране рисунков, состоящих из прямолинейных отрезков**.**

Последние два исполнителя были переписаны для системы 2.x недавно и сейчас находятся в стадии тестирования.

Планы

Поддержка материальных роботов в системе Кумир 2.1 (Lego EV3). Улучшения документации. Разработка обучающих практикумов. Расширение функций автоматической проверки заданий.

А. Г. Кушниренко, А. Г. Леонов, М. В. Райко
Москва, ФГУ ФНЦ НИИСИ РАН, МПГУ
http://www.mpgu.ru

Выравнивающий курс «Азы программирования» для первокурсников научно-технических и педагогических специальностей

Аннотация

В докладе будет описано содержание, организация и программно-методическое обеспечение выравнивающего семестрового курса по программированию для педагогических университетов. Курс включает 7 лекций и предполагает до 30 часов самостоятельной работы студентов в учебных системах ПиктоМир и КуМир, с автоматической проверкой результатов выполненных работ online.

Текущее состояние дел с изучением программирования в системе школьного образования РФ

В соответствии с Федеральным государственным образовательным стандартом основного общего образования (ФГОС ООО) в предметную область «Математика и информатика» входит курс информа-

тики. В учебном (образовательном) плане основного общего образования на изучение курса информатики отводится по 1 часу в неделю в VII-IX классах с общим количеством часов — 105.

Заметная часть из этих 105 часов теоретически должна была бы тратиться на изучение конкретных элементов программирования. Практически, в большинстве школ такого изучения не происходит и в результате, типичный выпускник общеобразовательной школы имеет серьезные пробелы в освоении общеобразовательных элементов программирования.

Требования нормативных документов к уровню освоения программирования школьниками РФ

Сам термин «элементы программирования» и выбор тех элементов, которые сегодня следует признать общеобразовательными, определяются действующими нормативными документами. К таким документам относится, в частности, утвержденная в 2015 году решением федерального учебно-методического объединения по общему образованию (протокол от 8 апреля 2015 г. № 1/15) ПРИМЕРНАЯ ОСНОВНАЯ ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА ОСНОВНОГО ОБЩЕГО ОБРАЗОВАНИЯ ПО ИНФОРМАТИКЕ. Эта программа предусматривает изучение материала, охватывающего базовые элементы программирования.

1 Извлечение из ПРИМЕРНОЙ ОСНОВНОЙ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ ОСНОВНОГО ОБЩЕГО ОБРАЗОВАНИЯ ПО ИНФОРМАТИКЕ в редакции от апреля 2015 года.

Алгоритмы и элементы программирования

Исполнители. Состояния, возможные обстановки и система команд исполнителя; команды-приказы и команды-запросы; отказ исполнителя. Алгоритм как план управления исполнителем (исполнителями). Алгоритмический язык (язык программирования) — формальный язык для записи алгоритмов. Программа — запись алгоритма на конкретном алгоритмическом языке. Компьютер — автоматическое устройство, способное управлять по заранее составленной

программе исполнителями, выполняющими команды. Программное управление исполнителем.

Алгоритмические конструкции

Конструкция «следование». Линейный алгоритм. Конструкция «ветвление». Условный оператор: полная и неполная формы. Выполнение и невыполнения условия (истинность и ложность высказывания). Простые и составные условия. Запись составных условий. Конструкция «повторения»: циклы с заданным числом повторений, с условием выполнения, с переменной цикла. Запись алгоритмических конструкций в выбранном языке программирования.

Разработка алгоритмов и программ

Оператор присваивания. *Представление о структурах данных.*

Константы и переменные. Переменная: имя и значение. Типы переменных: целые, вещественные, *символьные, строковые, логические.*

Табличные величины (массивы). Одномерные массивы. *Двумерные массивы.*

Примеры задач обработки данных:

- 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.

Знакомство с алгоритмами решения этих задач. Реализации этих алгоритмов в выбранной среде программирования.

Составление алгоритмов и программ по управлению исполнителями Робот, Черепашка, Чертежник и др.

Более широкий набор элементов программирования, выходящий за общеобразовательный минимум, но, по мнению авторов, не выходящий за пределы минимума, необходимого для студентов естественно-технических специальностей, приводится в другом нормативном документе — ежегодно обновляемом кодификаторе КИМ ЕГЭ по информатике.

2 Извлечение из Кодификатора элементов содержания и требований к уровню подготовки выпускников образовательных организаций для проведения единого государственного экзамена по информатике и ИКТ

Раздел 2. Перечень требований к уровню подготовки выпускников, достижение которого проверяется на едином государственном экзамене по информатике и ИКТ

Возможные алгоритмические задачи для подраздела 1.1 перечня требований к уровню подготовки выпускников, достижение которых проверяется на едином государственном экзамене по информатике и ИКТ.

- Нахождение минимума и максимума двух, трех, четырех данных чисел без использования массивов и циклов.
- Нахождение всех корней заданного квадратного уравнения.
- Запись натурального числа в позиционной системе с основанием, меньшим или равным 10. Обработка и преобразование такой записи числа.
- Нахождение сумм, произведений элементов данной конечной числовой последовательности (или массива).
- Использование цикла для решения простых переборных задач (поиск наименьшего простого делителя данного натурального числа, проверка числа на простоту и т.д.).
- Заполнение элементов одномерного и двумерного массивов по заданным правилам.
- Операции с элементами массива. Линейный поиск элемента. Вставка и удаление элементов в массиве. Перестановка элементов данного массива в обратном порядке. Суммирование элементов массива. Проверка соответствия элементов массива некоторому условию.
- Нахождение второго по величине (второго максимального или второго минимального) значения в данном массиве за однократный просмотр массива.
- Нахождение минимального (максимального) значения в данном массиве и количества элементов, равных ему, за однократный просмотр массива.

- Операции с элементами массива, отобранных по некоторому условию (например, нахождение минимального четного элемента в массиве, нахождение количества и суммы всех четных элементов в массиве).
- Сортировка массива.
- Слияние двух упорядоченных массивов в один без использования сортировки.
- Обработка отдельных символов данной строки. Подсчет частоты появления символа в строке.
- Работа с подстроками данной строки с разбиением на слова по пробельным символам. Поиск подстроки внутри данной строки, замена найденной подстроки на другую строку.

Выполнимы ли требования нормативных документов к уровню освоения программирования школьниками РФ, достигаемом в основной школе?

Ответ на этот вопрос безусловно положителен. При правильной организации и надлежащем оснащении школы программным и методическим обеспечением элементы программирования можно освоить, потратив на это не более трети из выделенных в основной школе на информатику 105 часов, то есть около 30–35 часов. За это время можно научить школьников устойчиво решать не только простейшие задачи обработки данных, перечисленные в разделе II.1 выше (в примерной программе по информатике основной школы), но и научить решать чуть более сложные задачи, перечисленные в кодификаторе ЕГЭ (раздел II.2).

Для интенсивного (с затратой всего лишь 30–35 учебных часов) освоения элементов программирования учениками 7–9 классов достаточно выполнение двух простых условий:

1. обеспечение доступа школьников на школьных (и, по возможности, на личных) компьютерах к системе программирования на одном из освоенных системой школьного образований РФ языков программирования: Паскаль, Бейсик, школьный алгоритмический, Питон;
2. установка в выбранной системе программирования комплекта минипрактикумов по программированию с автоматизированной

off-line проверкой правильности выполнения заданий. Практикумы должны содержать несколько десятков задач, включая как все алгоритмические задачи, перечисленные в кодификаторе ЕГЭ, так и подводящие к ним более простые задачи.

Эти два условия могут быть обеспечены разными путями.

Если в качестве языка программирования выбран «школьный алгоритмический язык», то

1) в качестве системы программирования и на школьных и на домашних компьютерах может использоваться свободно распространяемая многоплатформенная система программирования программирования КуМир и встроенные в эту систему механизмы подготовки практикумов с автоматизированной проверкой;

2) в качестве методического обеспечения могут использоваться КуМир-практикумы, разрабатываемые авторами настоящего доклада для первокурсников МПГУ, либо (для некоммерческого использования) разработанные известным учителем и методистом, доктором технических наук К.Ю. Поляковым, КуМир-практикумы *Робот, Массивы 1, Массивы 2, Строки* [1]. (Результаты использования перечисленных практикумов при переподготовке учителей информатики и начальных классов г. Сургута приведены в докладе [2].)

Проверочная программа составляется автором практикума на языке КуМир и включается в невидимую студенту часть задания. Одна программа может проверять несколько заданий (Пример приведен в Приложении.)

Если в качестве языка программирования выбран Паскаль, Бейсик или Питон, то номенклатуру заданий можно заимствовать из готовых КуМир-практикумов, а автоматизированную проверку можно организовывать вне выбранной системы программирования, например, с помощью свободно распространяемой программной системы e-judge.

Необходимость введения выравнивающего курса «Азы программирования» в педагогических университетах

Сегодня большинство учителей естественно-научных дисциплин (в том числе и большинство учителей информатики) не обладают устойчивыми навыками элементарного программирования. Поэтому неудивительно, что даже предусмотренные обязательной программой эле-

менты программирования, в России сегодня не осваиваются подавляющим большинством выпускников школы, в том числе и выпускниками, успешно поступающими на естественно-научные и педагогические специальности.

Действительно, анкетирование первокурсников мехмата МГУ за последние 10 лет показывает, что в каждой группе из 25 студентов обнаруживается от 3 до 6 студентов, которые не отладили в своей жизни ни одной программы на компьютере, а треть студентов не способна за 10 минут даже приступить к написанию программу нахождения числа максимальных значений в числовом массиве (на выбранном самим студентом языке программирования).

А опыт работы со студентами МПГУ показывает, что многие из них испытывают трудности даже в составлении простейших линейных и циклических программ, управляющих виртуальными исполнителями Робот, Черепашка, Чертежник.

А такие студенты не способны толком освоить многие университетские курсы, посвященные информационным технологиям, и получают диплом учителя (в том числе и учителя информатики), так и научившись программировать, что впоследствии сказывается на уровне освоения программирования их учениками.

Разорвать этот порочный круг можно путем введения в педагогических университетах выравнивающего курса небольшого объема, посвященного исключительно наработке технических навыков элементарного программирования, которые должны были бы быть освоены в средней школе.

Краткое описание курса «Азы программирования»

Курс расчитан на 14 часов лекций-демонстраций и 28 часов самостоятельных занятий (на университетских и домашних компьютерах) по выполнению практикумов с автоматизированной off-line проверкой с еженедельной on-line сдачей заданий.

На каждой из 7 лекций разбираются условия и примеры решения заданий двух практикумов, которые должны быть выполнены и сданы студентами в течение двух следующих недель. Весь справочный материал, необходимый для выполнения каждого задания содержится в самом задании. Дополнительные материалы размещаются в сети «В Контакте» в группе, открытой для слушателей курса. Через эту же группу студенты сдают выполненные задания и получают индиви-

дуальные консультации. Лектор курса берет на себя обязательство организовать ответы на вопросы студентов с задержкой не более 24 часов. Проверка правильности решений студентов автоматизирована — в каждое задание включена не видимая студентом проверяющая программа. Лектор может просматривать сданые студентами программы и при необходимости посылать студенту свои замечания.

Всего студенты выполняют 14 практикумов — 3 в системе ПиктоМир и 11 в системе КуМир:

1. Принцип программного управления виртуальными и реальными исполнителями, программа. Компьютер — исполнитель программ. Бестекстовая форма составления программы: повторители и подпрограммы. Система ПиктоМир.

2. Обратная связь при исполнении программ: условная и циклическая конструкции. Составление программ управления с обратной связью.

3. Использование счета в алгоритмах управления.

4. Текстовая форма составления программы. Система КуМир. Исполнитель Робот. Школьный алгоритмический язык. Запись повторителей, подпрограмм, условных и циклических конструкций в КуМире.

5. Практикум «Робот». (Решение задач по управлению Роботом с использованием целочисленных переменных в качестве счетчиков).

6. Практикум «Алгоритмы без циклов» (включает решения задач на нахождение минимума и максимума двух, трех, четырех данных чисел без использования массивов и циклов).

7. Практикум «Числовые последовательности» (однопроходные алгоритмы нахождения суммы, призведения, максимума, минимума, числа положительных элементов последовательности и массива).

8. Контрольная работа 1

9. Практикум «Массивы» (перестановка элементов массива в обратном порядке; циклический сдвиг элемента массива; вставка и удаление элемента; заполнение и суммирование двумерного масссива)

10. Практикум «Поиск и сортировка 1» (линейный поиск элемента в массиве и файле, пузырьковая сортировка, нахождение числа различных элементов массива)ю
11. Практикум «Поиск и сортировка 2» (бинарный поиск, слияние двух упорядоченных массивов в один без использования сортировки, сортировка слиянием).
12. Практикум «Строки» (обработка отдельных символов данной строки; подсчет частоты появления символа в строке; поиск подстроки внутри данной строки, замена найденной подстроки на другую строку)
13. Практикум «Последовательные файлы» (подсчет числа символов, числа слов, числа строк, замена группы пробелов на один пробел).
14. Контрольная работа 2.

Литература

[1] `http://kpolyakov.spb.ru/school/kumir.htm` проверено 22.01.2016.

[2] *А. Г. Кушниренко*, Пять практикумов К.Ю. Полякова по программированию с автоматизированной проверкой в системе КуМир. Десятая конференция Свободное программное обеспечение в высшей школе, Переславль, 24–25 января 2015 года. Москва, АльтЛинукс, 2015.

Приложение. Универсальная проверяющая программа для практикума «Робот» разработки К.Ю. Полякова

```
| Студент реализует в КуМире алгоритм с фиксированным именем Миссия
| При нажатии Ctrl-T этот алгоритм проверяется приведенным ниже кодом.
алг цел @тестирование
нач
  Миссия
  знач:=__оценка достижений Робота(10)
кон
алг цел __оценка достижений Робота(цел максимальный балл)
нач цел недокрашено, перекрашено;
  лит успех   = "Миссия выполнена успешно."
  лит неуспех = "Миссия не выполнена: "
  знач:=0
  _проверка правильности закраски(недокрашено, перекрашено)
  выбор
    при _Робот на базе и недокрашено=0 и перекрашено=0:
      вывод успех,нс; знач:=максимальный балл
    при _Робот не на базе:
```

```
      вывод неуспех,"Робот не пришел на базу.", нс;
    при недокрашено>0 и перекрашено=0:
      вывод неуспех, "не все нужные клетки закрашены.", нс
    при недокрашено=0 и перекрашено=1:
      вывод неуспех,"закрашена одна лишняя клетка.", нс
    при недокрашено=0 и перекрашено>1:
      вывод неуспех,"закрашены лишние клетки.", нс
    при недокрашено>0 и перекрашено>0:
      вывод неуспех,нс, "  не все нужные клетки закрашены, и", нс
      вывод             "  закрашены некоторые лишние клетки.", нс
  все
кон
алг лог _Робот на базе
нач цел x,y
  @@робот(x,y)
  знач:= @@нижняя буква(x,y) = 'Б'
кон
алг _проверка правильности закраски(рез цел недокрашено, рез цел перекрашено)
нач цел x,y,ширина,высота
  @@размер поля(ширина, высота); утв (ширина>0 и высота>0)
  недокрашено:=0; перекрашено:=0
  нц для x от 1 до ширина;
    нц для y от 1 до высота
      если @@метка(x,y) и не @@закрашена(x,y)
        то недокрашено:=недокрашено+1
      все
      если @@закрашена(x,y) и не @@метка(x,y)
        то перекрашено:=перекрашено+1
      все
    кц
  кц
кон
```

Леонов А. Г., Кузьменко М. А.
Москва, ФГОУ ВО «Московский государственный университет имени М.В.Ломоносова» механико-математический факультет
http://www.math.msu.ru, http://e.kumir.su, http://ejudge.math.msu.su/

Поколение «Reset» — «новые дети» на мехмате МГУ

Аннотация

В высшую школу приходит новое поколение школьников, не приученное к чтению длинных (как, впрочем, и коротких) текстов, самостоятельному преодолению технических и содержательных трудностей и многоминутной работе над одной и той же проблемой.

Вовлечение этих «новых детей» в интеллектуальные занятия требует новых подходов, учитывающих перечисленные особенности. Доклад основывается на опыте авторов по преподаванию программирования на мехмате МГУ в XX и XXI веке.

Традиционно на механико-математическом факультет МГУ, в начале семестра студентам I курса предлагается ответить на простые вопросы и решить несколько простых задач, для того, чтобы преподаватель получил представление об уровне знаний и умений в области Информатики и математики:

8. В школе меня: *не учили программировать, учили программировать на языке .....*

9. К моменту окончания школы я *вообще не научился программировать, немного научился программировать, неплохо научился программировать, считался в своем классе экспертом по программированию и компьютерам*

10. (5 мин.) Допишите программу вычисления числа различных корней уравения $x(x-a)(x-b)=0$

```
алг число корней(вещ a,b)
нач цел n
если (a=0.) то n:=1 иначе n:=2 все
утв    | n равно числу корней уравнения x(x-a)=0
если (b<>0. и ...    ) то ...    | учтен корень b
вывод "число корней равно ", n
кон
```

Как видно, задачи не представляют сложности для современного школьника, знакомого с азами алгоритмизации (успешно сдавшим ЕГЭ по Информатики и ИКТ). Среди задач есть и простейшая «подсчитать число разных из трех (a, b, c)». Ответ нужно дать на любом языке программирования, включая русский. При этом общая тенденция, замеченная авторами, прослеживаемая в течение последних 15 лет, это увеличение количества школьников, которые НЕ в состоянии изложить НИКАК последовательность действий для решения поставленной задачи. Грубо говоря, число вчерашних школьников, не способных решить эту задачу все возрастает. Хотя, при этом, студенты легко могут дать ответ, на конкретный вопрос, например: «Число разных из 3, 4, 4”.

К сожалению, алгоритмическое мышление не дается человеку с рождения. Еще в XX веке основатель школьной Информатики акаде-

мик А.П. Ершов строил модель выпускника эпохи информационного общества, понимая под моделью совокупность знаний, умений и навыков, которыми должен владеть выходящий из школы современный молодой человек. А.П. Ершов предложил выбрать программиста как человека, который в существенно большей степени, чем любой другой специалист, всей своей повседневной как рутинной, так и творческой деятельностью формировал в себе умения и навыки, помогавшие ему наиболее эффективно использовать вычислительную технику [1]: «Программист обязан обладать способностью первоклассного математика к абстракции и логическому мышлению в сочетании с эдисоновским талантом сооружать все, что угодно из нуля и единицы, он должен соединять в себе аккуратность бухгалтера с проницательностью разведчика, фантазию автора детективных романов с трезвой практичностью бизнесмена, а, кроме того, иметь вкус к коллективному труду, быть лояльным к организатору работ и т.д... Программист — солдат второй промышленной революции и как таковой должен обладать революционным мышлением и мужеством».

Современные выпускники школ, как жители информационного общества, существуют в множестве непрерывных потоков информации (телевизор, Интернет и т.п.), при которых индивидууму нужно оперативно реагировать практически на все внешние раздражители. В качестве таких раздражителей выступают и преподаватели университета. Неумение справляться с всё возрастающей информационной нагрузкой, часто отбрасывает (по информационной ценности) занятия в университете на второе место.

Еще 25 лет назад, материал, излагаемый преподавателем на занятии, предлагался студентам в виде печатных материалов. Авторы заметили, что на рубеже веков, студенты стали терять (в прямом смысле этого слова) часть материалов. Наличие Интернета и электронной почты, как средства коммуникации, не внесло значительных изменений. Ни электронная почта, ни сервер, с материалами занятий не являлся источником информации для студентов, так как они не выступают в качестве ежедневого «раздражителя», в отличии, например, от социальных сетей.

Размещение материалов в сети «В Контакте» привело к увеличению интенсификации занятий по программированию. Во-первых, все вновь размещенные материалы, автоматически попадают в «новости». Во-вторых, преподаватель получил «живой» канал общения со студентами.

Объем задач, которые студент может решить в течение семестра ограничен не только возможностями (знаниями и навыками) студента, но и возможностями преподавателя эффективно контролировать выполнение студентами заданий.

В качестве промежуточного решения было внедрена автоматическая проверка заданий студентов в системе Ejudge [2]. Возможность решения и сдачи заданий не только непосредственно преподавателю, но и самостоятельно, дистанционно, с помощью автоматизированной проверки, позволило существенно увеличить количество задач в семестре. Сдача автоматизированной системе для молодого поколения превратилось в аналог прохождения квеста, в своеобразную, увлекательную игру, что не только повысило мотивацию студентов к изучению программирования, но и позволило преодолеть психологический барьер перед сложными заданиями.

Авторы не утверждают, что фронтальные методы обучения изжили себя, но, при этом очень надеются, что и «новые дети» которые придут на мехмат МГУ в 2016 году не принесут им существенных новых проблем.

Литература

[1] *Ершов А. П.* О человеческом и эстетическом факторах в программировании // Кибернетика. — №5, — 1972.

[2] `http://ejudge.math.msu.su/`

Бесшапошников Н. О., Ерёмин Д. Б., Кушниренко А. Г., Леонов А. Г., Райко М. В., Усова Е. Р.
Москва, ФГУ ФНЦ НИИСИ РАН
`http://www.piktomir.ru`

ПиктоМир — 2016

Аннотация

В докладе будет продемонстрирована мобильная реализация бестекстовой учебной системы программирования ПиктоМир, обеспечивающая индивидуальную и коллективную работу по составлению программ управления виртуальными и реальными роботами.

Вот уже 2 года существует в Нидерландах проект под названием «Steve Jobs Schools»[1]. В пилотных школах королевства учителей «заменили» на планшеты iPad. Минусы этого спорного проекта общеизвестны, однако есть и позитивные стороны. Планшетные компьютеры для детей, обучающихся в младшей школе или в дошкольном образовательном учреждении, гораздо удобнее в использовании, чем лаптопы и настольные ЭВМ. Эргономические параметры и интерфейс планшетов комфортнее для ребенка.

В течении последних лет в детских садах и школах Москвы и России проходит эксперимент по обучению программированию детей 5-7 лет. Для этого педагоги используют новационный программный продукт — ПиктоМир . Сейчас уже можно с уверенностью констатировать, что ПиктоМир прошел успешную апробацию и зарекомендовал себя как удобное средство для обучения детей азам последовательного программирования [2].

В ПиктоМире учащийся видит знакомого по мультфильмам и кинофильмам персонажа — Робота. Робот для ребенка понятен и прост. Его даже нечего изучать. Ребенок привык управлять героями в компьютерных играх, поэтому ассоциируя себя с Роботом, умеющим выполнять простейшие действия (шагать вперед, поворачиваться и «красить» — ремонтировать площадку, по сценарию игры) управлять им используя кнопки на экране планшета (так называемое, ручное управление) для ученика привычно: нажимаешь на кнопки пульта и Робот выполняет команды играющего. Да и задача, которую требуется решить — тривиальна.

Можно предложить детям кооперативную игру, в которой, один ребенок, назовем его Командиром, дает команды другому ребенку-Роботу. Ученик-Робот умеет выполнять только элементарные действия, именно такие, которые выполнял Робот на экране.

Задача детям ставится та же самая: отремонтировать поле космодрома. Воображаемый космодром (лабиринт) можно легко изготовить при помощи бумаги и фломастеров.

Игра у детей начинается с договора. Дети договариваются о начале игровой деятельности, совместно составляют карту Космодрома, распределяют между собой роли и выстраивают свои действия и поведение в соответствии с выбранной ролью.

Взяв на себя роль, ребенок начинает принимать и понимать четкость ролевых прав и обязанностей. Так, например, Командир, который управляет Роботом, требует от игрока-Робота сделать шаг впе-

ред, повернуться, отремонтировать клетку Космодрома, то есть потребовать, чтобы Робот четко выполнял его указания.

Действуя с предметами-заместителями (поле Космодрома, испорченные или отремонтированные клетки), ребенок начинает оперировать в мыслимом, условном пространстве. Предмет-заместитель становится опорой для мышления. Постепенно игровые действия сокращаются, и ребенок начинает действовать во внутреннем, умственном плане. Таким образом, игра способствует тому, что ребенок переходит к мышлению в образах и представлениях.

В продолжении игры, учитель немного меняет условие, рассказывая, что Робот находится на далекой-далекой планете, радио-сигнал до нее идет дни и месяцы, поэтому Командир не может командовать непосредственно. Это попросту невозможно. Командир пишет последовательность действий на листочке бумаги. А игрок-Робот, читая записи, последовательно выполнят их. И ребята вместе проверяют правильность составленных команд — программу. Итак, программирование начинается.

Игру с правилами можно перенести уже на виртуального Робота. Например, по выбору педагога дети объединяются в группу. Каждый ребенок со своего планшета, самостоятельно, программирует своего Робота, для решения поставленной перед ним задачи. Учащийся может индивидуально выполнить свою программу, посмотреть на результат, отредактировать составленную программу. Когда, программа готова, ребенок отправляет ее своим партнерам по игре (запускает игру). Таким же образом он получает результаты программирования товарищей. При этом оценивается не только успешность решения поставленной задачи, но и, например, время, затраченное ребенком на решение задачи.

В этих состязательных играх всегда появляются лидеры, которые быстрее и легче выполняют поставленные задачи, осваивают новую обстановку, нового Робота. Стихийно появившиеся лидеры с удовольствием контролируют отстающих детей, помогая им с составлением программы, освоением нового Робота. Эти процессы происходят под наблюдением педагога, но без непосредственного участия последнего в процессе обучения. Эффективность кооперативного обучения подчас сильно превышает индивидуальные занятия для дошкольников.

Кооперативные (как и состязательные игры) с одной стороны позволяют всем ученикам активно участвовать в изучении программирования, с другой стороны являются сильным стимулом к развитию

алгоритмического мышления, так как дети стараются победить в игре и(или) совместными усилиями решить поставленную задачу.

Поскольку в игру включены несколько планшетных компьютеров, то для их функционирования необходима локальная сеть. Для работы в сети используется специальная, сетевая часть системы ПиктоМир.

Задача сетевой части ПиктоМира обеспечить взаимодействие между учениками в классе и позволить преподавателю посредством его устройства следить за работой учеников, их прогрессом, давать задания и т.п.[3]

Не существует отдельно управляющего ПиктоМира (учительского) и игрового (ученического). Эти роли исполняет одно и тоже приложение. С точки зрения ученика приложение ПиктоМир всегда находится в режиме «Ученик». Сетевое общение начинается с автономной работы «Учеников», как ведомых устройств. Эти приложения позволяют автономно составлять программы для Роботов, находясь одновременно, в состоянии ожидания появления «Учителя», распорядителя. «Учитель» — это режим приложения ПиктоМир, в который только преподаватель может перевести свое устройство. Для этого используется аппарат аутентификации.

Для упрощения администрирования принято, что в одной сети не может быть более одного учителя, в таком случае при активации «Учителя», ведомые устройства («Ученики») находят и подключаются к нему автоматически.

Для реализации этой функции каждое неподключенное ведомое устройство с заданным интервалом отправляет широковещательное сообщения поиска «Учителя». «Учитель» получив такое сообщение отправляет в ответ «Ученику» свои данные для подключения.

Даже в режиме индивидуальной (неавтономной) работы, у преподавателя есть возможность наблюдать со своего планшета (или компьютера) за прогрессом в решении задач учениками, в том числе, одновременно за всеми.

Учитель может ставить каждому отдельному ученику индивидуальное задание без непосредственного взаимодействия педагога с планшетом ребенка.

В режиме многопользовательских игр, чтобы снизить нагрузку на устройство «Учитель», для каждой группы учеников случайным образом выбирается мастер-устройство каждой конкретной игры, и ему отправляется информация для коллективной игры, подключения друг к другу, как бы создавая временную подсеть на игру. «Учитель»

при этом поддерживает соединение и собирает статистику только с мастер-устройств таких подсетей. Кооперативные игры в ПиктоМире не только помогают учащимся эффективнее осваивать основы программирования, но и упрощают работу педагогу в классе , позволяя планировать и распределять задания и контролировать процесс обучения.

Литература

[1] O4NT foundation `http://stevejobsschool.nl/nieuw-onderzoek-kinderen-worden-creatiever-van-de-ipad-op-school` // (дата обращения: 24.01.2016).

[2] *Кушниренко А. Г., Леонов А. Г., Пронин К. А., Ройтберг М. А., Яковлев В. В.* «Свободное программное обеспечение в высшей школе», 29–30 января // ПиктоМир: опыт использования и новые платформы. — Переславль, 2011.

[3] *Леонов А. Г.* Тенденции объектно-ориентированного программирования в разработке системы КуМир// Программные продукты и системы, Тверь, 2012 — № 4 с. 245–249

А. Г. Кушниренко, А. Г. Леонов, М. А. Ройтберг
Москва, ФГУ ФНЦ НИИСИ РАН
`http://www.piktomir.ru`, `http://www.kumir.su/`

Алгоритмика для дошкольников и младших школьников

Аннотация

Современные информационные технологии стали обычным элементом общей культуры дошкольников и школьников. В докладе предложен один из возможных путей освоения начальной алгоритмической грамотности в младших классах и дошкольных учреждениях. В качестве педагогических программных средств предлагаются системы ПиктоМир и КуМир.

Сегодня, в первой четверти 21 века, на каждого жителя Земли приходится несколько микропроцессоров, число переключательных элементов в каждом из которых приближается к числу нейронов в

человеческом мозге. Построенные на базе таких микропроцессоров компьютеры начинают превосходить человека в тех областях, которые ранее считались не поддающимися автоматизации: игра в шахматы, узнавание человека по фотографии, вождение автомобиля, игра «что-где-когда», финансовые спекуляции.

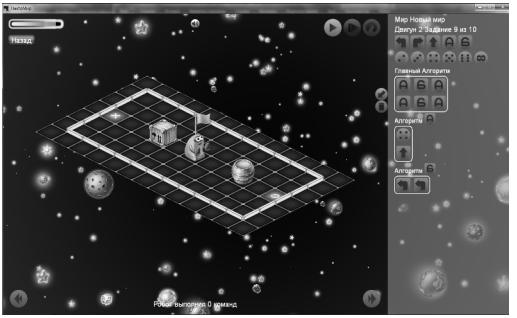
Экономисты и социологи говорят, что человечество переходит от индустриального уклада к информационному.

Неудивительно, что под воздействием этих радикальных изменений в образе жизни человечества меняется и традиционное понятие грамотности. Заголовок одной из статей в газете Нью Йорк Таймс — «Чтение, письмо, счет, а теперь и программирование» — метко описывает суть происходящих изменений [1].

По поводу новой грамотности — алгоритмической — сегодня даже намека на такой консенсус нет ни в профессиональном педагогическом сообществе, ни в родительских кругах, ни среди работодателей в сфере информационных технологий. Авторы доклада попытаются описать тот уровень всеобщей алгоритмической грамотности, достижение которого к концу первого-второго года школьного обучения реально было бы обеспечить и стоило бы обеспечить в России.

Мы предлагаем организовать постижение начальной алгоритмической грамотности в три этапа-уровня. Уровни 1 и 2 могут быть освоены в подготовительной группе детского сада, уровень 3 — в первом классе начальной школы. На освоение дошкольных уровней 1 и 2 мы отводим суммарно 24 занятия по 30 — 40 минут каждое и на освоение школьного уровня 3 — еще 12 занятий по 45 минут. Первая половина каждого занятия — бескомпьютерная. Вторая половина каждого занятия посвящается индивидуальному или кооперативному составлению программ по управлению виртуальными и реальными роботами. Хотя без реальных роботов на уровнях 1 и 2 можно и обойтись, их использование радикальным образом улучшает мотивацию и глубину освоения материала.

На уровнях 1 и 2 программы составляются на планшетах на бестекстовом (пиктограммном) языке программирования, доступном дошкольникам-шестилеткам.



На уровне 3 используется текстовый язык с национальной лексикой, требуемое подмножество которого доступно первоклассникам-семилеткам. В это подмножество не входит понятия: переменная, выражение, операция присваивания, а числовые значения-константы используются только в качестве повторителя и как аргументы команд виртуального или реального робота. Для решения ряда задач управления роботами в качестве суррогата целочисленной переменной во внешнюю обстановку вводится исполнитель Счетчик.

В опробованных с участием авторов методиках используются языки программирования разработанных в НИИСИ РАН учебных программных систем ПиктоМир и КуМир [2-6]. На школьном уровне Пиктомир и Кумир методически представляют собой единую систему: можно начать составлять пикто-программу в ПиктоМире, затем автоматически перевести ее в текстовую форму и продолжить работу в КуМире.

По мнению авторов, начальный курс алгоритмической грамоты естественно разбивается на следующие три уровня.

Уровень 1 (12 занятий).

1.1. Парадигма программного управления исполнителями. Понятия:

- робот — исполнитель команд; система команд исполнителя; обстановка, в которой «работает» исполнитель; возможность аварии при исполнении данной команды в данной обстановке;
- алгоритм — пошаговый план будущих действий по управлению исполнителем с целью достижения определенной цели;

- исполнение алгоритма — процесс последовательной выдачи команд исполнителю в соответствии с заранее выработанным планом;
- программа — алгоритм, представленный в такой форме, которая позволяет поручить исполнение алгоритма компьютеру или другому автоматическому устройству;
- разделение обязанностей: робот — исполнитель команд, компьютер — исполнитель программ; программист — составитель программ;
- язык программирования — конкретный набор правил составления программ для исполнения компьютерами определенного типа.

1.2. Правила составления программ (без обратной связи) в пиктограммном языке программирования: повторители и вспомогательные алгоритмы.

1.3. Реальный робот и его виртуальный партнер.

Сборка (из готового робототехнического комплекта, например Lego EV3) реального исполнителя-робота без обратной связи. Расхождение результатов выполнения отдельных команд и последовательностей команд реальным и виртуальным роботами. Составление пиктопрограмм для управления без обратной связи реальным роботом и его виртуальным партнером (рекомендуется виртуальный робот из компьютерной игры Сокобан).

Уровень 2. (12 занятий)

2.1. Методика коллективного выполнения одной работы двумя (несколькими) программистами: этап деления общей работы на две части, этап утверждения договоренности о разделе работы, этап составления и отладки программ (компьютеры партнеров контролируют соблюдение утвержденных договоренностей).

2.2. Команды-вопросы. Обратная связь при управлении исполнителем.

Возможность придумывания одного алгоритма с обратной связью, позволяющего достигать аналогичных целей для серии аналогичных обстановок.

Правила составления программ (с обратной связью) в пиктограммном языке программирования: цикл «пока» и ветвление.

2.3. Придумывание алгоритмов и составление программ с обратной связью

2.4. Примеры алгоритмов управления роботом-исполнителем, требующие подсчета числа шагов. Исполнитель «счетчик». Алгоритмы и программы последовательного управления несколькими исполнителями с использованием обратной связи.

2.5. Параллельное управление несколькими однотипными исполнителями с помощью одной и той же программы без обратной связи.

Уровень 3. (12 занятий)

3.1. Пиктограммные (детские) и текстовые (взрослые) языки программирования. Демонстрация автоматического перевода программ управления виртуальным исполнителем с пиктограммного языка на текстовый.

3.2. Правила перевода программ из пиктограммного представления в текстовое:

- общие правила текстовой записи программ: запись команд управления исполнителями, освоенными на этапе 1; запись нескольких команд в одной строке;
- синтаксические ошибки в программах на текстовых языках, правила их обнаружения и исправления;
- правила записи цикла «пока» и ветвлений;
- «заклинания» для использования целочисленной переменной в качестве счетчика (например, в языке КуМир: цел а; а:=а+1; а:=а-1; а=0; а<>0.)

3.3. Сборка (из готового робототехнического комплекта, например, Lego EV3) реального исполнителя-робота с командами обратной связи и составление простейших программ управления реальным роботом с обратной связью.

Достоинства КуМира. Русская лексика, непрерывная индикация синтаксических ошибок. Возможность задания шаблона программы, защищенного от случайного искажения. Возможность организации автоматической проверки правильности выполнения заданий. Совместимость с ПиктоМиром. Возможность использования языка и системы в курсе информатики основной школы и при подготовке к ГИА.

Достоинства методики. Легкость освоения воспитателями ДОУ, не имеющими специальной подготовки. Большой объем (до 50%) некомпьютерных коллективных активностей. Отсутствие необходимости хранить результаты работы учеников от занятия к занятию. Интеграция работы с виртуальными и реальными роботами. Компьютер-

ная поддержка процедур кооперативной работы по составлению программ.

Литература

[1] `http://www.nytimes.com/2014/05/11/us/reading-writing-arithmetic-and-lately-coding.html`

[2] *Rogozhkina I. B., Kushnirenko A. G.* PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment. — // World Conference of Educational Technology and Researches, July, 2011

[3] *Кушниренко А. Г., Леонов А. Г.* Программирование для дошкольников и младших школьников. — // Информатика. — М.: Первое сент., 2011, N15. — стр. 20–23

[4] *Кисловская А. Д., Кушниренко А. Г.* Методика обучения алгоритмической грамоте дошкольников и младших школьников — // Информационные технологии в обеспечении федеральных государственных образовательных стандартов: Материалы Международной научно-практической конференции. 16–17 июня 2014 года. — Елец: ЕГУ им. И. А. Бунина, 2014. — Т. 2. — стр. 3–7.

[5] *Яковлев В. В.* «ПиктоМир: опыт использования и новые платформы», презентация к выступлению на 6-ой конференции «Свободное программное обеспечение в высшей школе», январь 2011, Переславль-Залесский, `http://www.gosbook.ru/node/32747`

[6] *Яковлев В. В.* Кумир 2.0. Компилятор и среда выполнения (доклад на OSEDUCONF-2013), `http://talks.rosalab.com/Kumir-20`

Иван Евгеньевич Панченко
Москва, Постгрес Профессиональный

Как мы создавали портал ВШЭ в 2002-2012 гг.

Аннотация

Автор доклада в эти годы работал в Группе компаний Стек, которая участвовала в разработке нескольких образовательных порталов, и затем, по заказу ВШЭ, разрабатывала портал этого ВУЗа. В работе использовались свободно-распространяемые технологии, и собственные разработки Стека. В докладе будет рассказано о том, какие трудности приходилось преодолевать и как, как была организована работа, как эволюционировали представления о том, каким должен быть портал и почему.

В конце 1999 года автор присоединился к команде сотрудников МГУ, разрабатывавших систему научно-популярных порталов «Научная Сеть». Выбор возможных технологий тогда был невелик, однако, понимая, насколько будущее проекта зависит от правильно выбранного технологического стека, мы проделали существенную работу по формированию программно-технологической платформы, получившей название Discovery. В основу Discovery легли свободно распространяемые продукты – СУБД PostgreSQL, Web-сервер Apache, сервер приложений mod_perl, Web-движок HTML::Mason. Данная платформа нашла затем применение в проектах Рамблера, и в 2002 году её обновленный вариант X-Ware, содержащий ORM-систему, асинхронную очередь транзакций, ролевую систему управления доступом, и ряд других полезных модулей, лег в основу проектов образовательных порталов, разрабатывавшихся группой компаний «Стек» в рамках государственного проекта единой образовательной среды.

Эта среда состояла из «горизонтальных» и тематических порталов. Автору довелось участвовать в создании пяти из них. В одном из этих проектов, наиболее интересном и жизнеспособном, головным исполнителем была Высшая Школа Экономики (ВШЭ). Типовая структура порталов диктовалась госзаказчиком, основой ее была идея каталогизации и поиска различных учебных материалов, находящихся в интернете, библиотеках и других источниках. Другую идею выдвигал зам. декана факультета бизнес-информатики А.П. Сериков, предлагавший отталкиваться от потребностей и структуры учебного процесса, собирая информацию, необходимую для конкретных учебных кур-

сов, и развивать интерактивные сервисы, что дало бы возможность эффективно использовать портал в учебном процессе. В дальнейшем эта идея частично была воплощена в «интернет-курсах», организованных на портале проректором ВШЭ В.В. Радаевым. Позже именно это направление развития было воплощено LMS-системами.

Следующим этапом стало создание собственного портала ВШЭ. Он создавался поэтапно с 2006 года, вначале как интернет-портал, ориентированный в основном на внешнюю аудиторию, получила развитие и внутрикорпоративная («интранет») составляющая, постепенно формирующая информационную архитектуру ВУЗа. Параллельно с порталом, взаимодействуя с ним, развивались другие информационные системы разного происхождения. При этом постоянно имела место конкуренция и взаимодействие между свободно распространяемыми и проприетарными решениями.

Литература

[1] *Губанов В. С., Лысаков С. В., Плечев П. Н.* Технологические возможности поддержки сервисов сети порталов / Сб. научн. ст. «Интернет-порталы: содержание и технологии». Вып. 1. ГНИИ ИТТ «Информика». — М.: Просвещение, 2003. — с. 421–433.

[2] *Добрякова М. С., Радаев В. В.* Образовательный портал по экономике, социологии и менеджменту // Социологические исследования. 2004. № 1. с. 131–133.

Михеев Андрей Геннадьевич
Москва, RunaWFE
http://runawfe.org/

Обучение процессному управлению: Работа со слоем данных

Аннотация

Сложным моментом обучения разработке бизнес-процессов, предполагающих исполнение экземпляров в компьютерной среде предприятия, является взаимодействие бизнес-процессов с данными в случаях, когда процессное управление преподается студентам, обучающимся по специальности бизнес-информатика или финансово-бухгалтерским

специальностям. Студенты этих специальностей не обладают необходимыми для практической работы с данными знаниями теории баз данных. В докладе представлен опыт обучения студентов методам упрощенной работы с данными, реализованным в свободной системе управления бизнес-процессами предприятия RunaWFE, полученный в НИТУ МИСиС.

Бизнес-процессы и бизнес-объекты

В соответствии с процессным подходом деятельность предприятия представляется в виде множества выполняющихся экземпляров бизнес-процессов. При этом состояние всего предприятия на определенный момент времени определяется состоянием всех бизнес-объектов предприятия на этот момент времени. Совокупность всех бизнес-объектов предприятия называется слоем данных. Процессный подход предполагает, что состояния бизнес-объектов изменяются только экземплярами бизнес-процессов. Здесь можно использовать аналогию с бухгалтерским учетом: бизнес-объекты будут соответствовать счетам бухгалтерского учета, а бизнес-процессы — проводкам.

Системы управления бизнес-процессами (СУБП) автоматизируют исполнение бизнес-процессов: В соответствии со схемой бизнес-процесса они раздают задания исполнителям и контролируют их выполнение.

Бизнес-объекты хранятся в других системах. Традиционно в качестве хранилищ для бизнес-объектов используются системы управления контентом (ECM-системы), ERP-системы или системы управления базами данных.

Так как взаимодействие бизнес-процессов с бизнес-объектами является важным аспектом процессного управления, в учебных курсах по процессному управлению надо формировать у студентов практические навыки организации такого взаимодействия. Однако, студенты финансово-бухгалтерских специальностей, изучающие процессное управление, как правило, не знакомы с теорией реляционных баз данных. Системы класса ERP или ECM им также не преподаются в объеме, позволяющем настраивать коннекторы к таким системам. Изучению таких систем, а также изучению теории баз данных посвящены серьезные курсы обучения, заметно превосходящие по объему курс процессного управления.

Создание слоя бизнес-объектов для учебных целей

В качестве хранилища бизнес-объектов в учебных целях при изучении процессного управления предлагается использовать листы документов Microsoft Excel. Это позволяет студентам финансово-бухгалтерских специальностей пользоваться уже известным им инструментом. Кроме того, при выполнении заданий на домашнем компьютере студентам не требуется устанавливать и настраивать сложные системы для работы с данными. LibreOffice Calc или MS Excel, как правило, уже установлен на компьютере пользователя.

Данные предлагается хранить на листах MS Excel в виде таблиц. Одна таблица соответствует одному листу документа MS Excel.

Для работы с данными в бизнес-процессе надо создать составной тип переменной, типы и названия полей которой будут соответствовать типам и названиям столбцов таблицы. Выборка строк из таблицы будет соответствовать списку переменных составного типа.

Работа с данными осуществляется при помощи специального бота (автоматического исполнителя заданий). В задачах боту используется четыре команды:

- INSERT
- SELECT
- UPDATE
- DELETE

В качестве параметров в этих задачах используются переменные бизнес-процессов.

Для работы с данным хранилищем используется специальный обработчик «Внешнее хранилище данных» (`ru.runa.wfe.office.storage.handler.ExternalStorageHandler`), позволяющий выполнять простейшие действия с данными:

Команда INSERT

Предназначена для добавления данных в таблицу. Вставка выполняется в следующую свободную строку. В качестве входных данных используется переменная составного типа или список из таких переменных.

Команда SELECT

Предназначена для чтения данных из таблицы, может быть использовано с условием. Например, для выборки заявок с определен-

ным статусом. В качестве результата всегда возвращает список, даже если был получен только один элемент.

Команда UPDATE

Предназначена для обновления переменной пользовательского типа в таблице. В параметрах этой команды обязательно надо использовать условие для определения переменной, которую требуется обновить.

Команда DELETE

Предназначена для удаления строки (строк) таблицы. В параметрах этой команды обязательно надо использовать условие для определения строк, которые необходимо удалить.

Для того, чтобы избежать конфликтов при одновременном изменении данных несколькими экземплярами бизнес-процессов, в системе RunaWFE, которая используется для проведения практикума, был реализован режим последовательной обработки заданий бота. Задания боту могут ставиться в очередь как на уровне задания, так и на уровне бота.

Реализация взаимодействия бизнес-процессов с бизнес-объектами в свободной системе RunaWFE

Покажем, как осуществляется взаимодействие со слоем данных в системе RunaWFE на примере бизнес-процесса заказа автотранспорта. Основные шаги бизнес-процесса: Сотрудник подает заявку, содержащую типа автомобиля, дату — время подачи и количество мест. Руководитель одобряет заявку или отказывает. После одобрения, заявка сохраняется во внешнем хранилище. Диспетчер выполняет отбор заявок на текущий день. После совершения поездки заявка получает статус «Исполнена».

На Рис. 1 представлено создание составного типа переменных для заявки на автотранспорт.

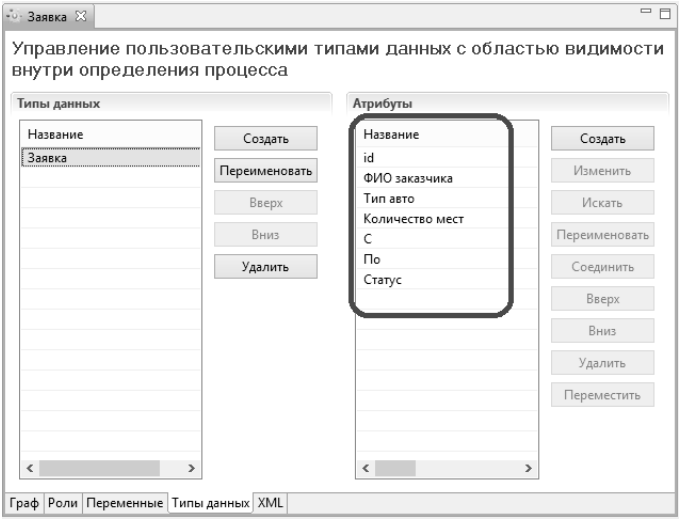


Рис. 1: Создание составного типа переменных

На Рис. 2 изображена таблица, расположенная на MS Excel листе, соответствующая созданному типу переменных «Заявка».

	A	B	C	D	E	F	G
1	7	Иванов И.И.	внедорожник	5	29.06.2015 15:55	29.06.2015 15:55	Не обработана
2	9	Петров А.Б.	газель	10	29.06.2015 15:55	29.06.2015 15:55	Не обработана
3	10	Сергеев М.У.	автобус	15	29.06.2015 15:55	29.06.2015 15:55	Не обработана
4	11	Сидоров В.Г.	лимузин	8	29.06.2015 15:56	29.06.2015 15:56	Не обработана
5							

Рис. 2: Таблица, содержащая заявки на заказ автотранспорта

Для работы с заявками требуется создать бота для взаимодействия с хранилищем данных. На Рис. 3 показана конфигурация бота для задачи добавления заявки на автотранспорт в таблицу. В конфигурации указывается выполняемая команда («INSERT»), путь к MS Excel файлу, содержащему таблицу («C:\runa_tmp\DB.xlsx»), атрибут, содержащий значение, которое будет связано с переменной бизнес-процесса («Заявка»), номер страницы в файле MS Excel, на

которой находится таблица и номер столбца, начиная с которого располагается таблица.

Конфигурации для действий SELECT, UPDATE и DELETE отличаются только названием команды и наличием условия для выбора строк.

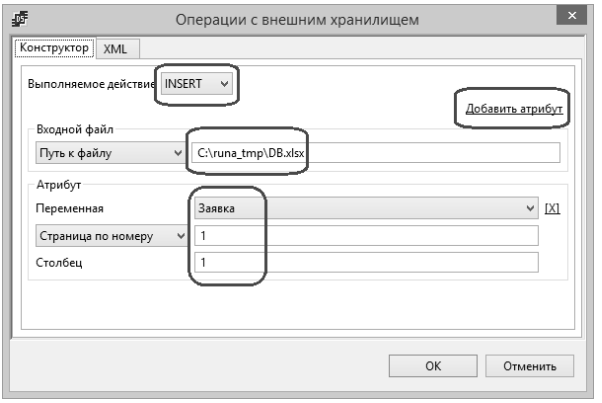


Рис. 3: Конфигурация для задачи добавления заявки на автотранспорт в таблицу.

В случае, если несколько пользователей одновременно вносят заявки на автотранспорт, может возникнуть конфликт доступа к данным. Для того, чтобы избежать таких ситуаций, в системе RunaWFE, была добавлена возможность синхронизации исполнения заданий ботами. В режиме синхронизации бот может обрабатывать свои задания только последовательно, друг за другом. Таким образом исключается одновременный доступ нескольких бизнес-процессов к одному файлу, если работа с данными MS Excel файла происходит при помощи только одного бота.

На Рис. 4 приведена форма, в которой можно сделать такую настройку.

Заключение

В настоящем докладе представлен опыт обучения, полученный в НИТУ МИСиС. Методику обучения легко перенести в другие ВУЗы,

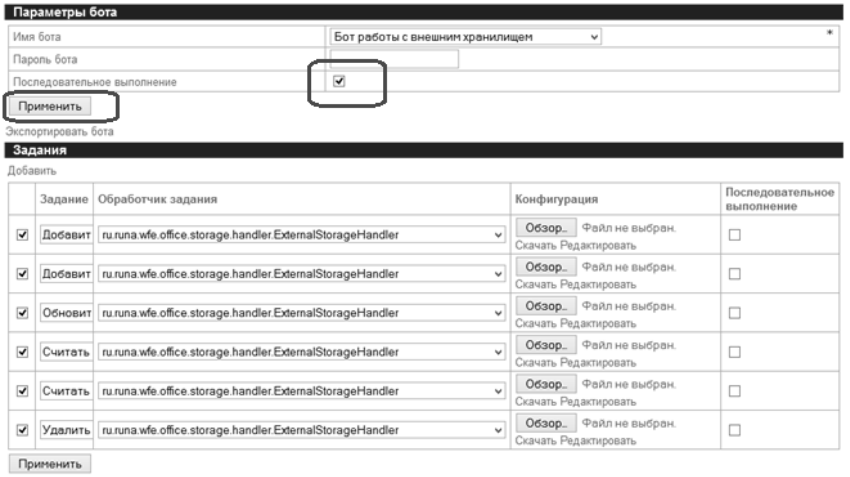


Рис. 4: Форма настройки последовательного выполнения задач ботом

так как она построена на использовании свободного программного обеспечения, доступного через интернет, не требующего оплаты или регистрации (система RunaWFE [1]). Практическое занятие по работе с внешним хранилищем данных вошло в курс, опубликованный на ресурсе Intuit.ru [2].

Литература

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

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

[3] Ссылка на сайт проекта RunaWFE: `http://runawfe.org/`

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

Универсальное средство для работы с SQL и NoSQL базами данных СПО DBeaver

Аннотация

Представлены возможности универсального бесплатного менеджера баз данных DBeaver для использования разработчиками и администраторами баз данных, а также для обучения студентов ИТ-направлений приемам работы с SQL и NoSQL базами данных

В настоящее время при создании приложений, особенно для обработки разнородной информации необходимо использовать базы данных различного типа, такие как SQL и NoSQL (не только SQL). Многие из таких баз данных являются СПО (свободным или свободно распространяемым программным обеспечением), что делает их доступными для использования широким кругом специалистов, а также и для обучения студентов. Однако до недавнего времени для работы с любой СУБД необходимо было устанавливать соответствующую графическую оболочку (например, MySQL Workbench для СУБД MySQL, Robomongo для СУБД MongoDB), либо использовать командную строку.

Наличие универсального инструмента для разработчиков и администраторов баз данных позволило бы оптимизировать их деятельность. Кроме того, такой универсальный инструмент был бы полезен при обучении студентов различных направлений ИТ-подготовки, поскольку позволил бы студентам освоить работу с различными СУБД через единый графический интерфейс. Более того, программные средства, используемые для обучения студентов, должны быть доступны, то есть быть СПО и обладать при этом необходимыми функциональными возможностями.

Мультплатформенный универсальный (работает с большим числом СУБД, как SQL, так и NoSQL) менеджер баз данных DBeaver как раз и является инструментом, который в значительной степени облегчает проектирование, реализацию, модификацию и работу с

различными СУБД. DBeaver является СПО и распространяется под лицензией GNU GPL2, написан на языке Java, в его основе лежит платформа Eclipse, совместим практически со всеми операционными системами семейств Windows, Linux, Mac OS, Solaris (x86), поскольку имеются 32-х и 64-х битные версии. Используются два варианта DBeaver, а именно: Community editions и Enterprise editions. Enterprise Edition включает поддержку СУБД NoSQL [1].

Для установки достаточно выбрать вариант (Community editions или Enterprise editions), скачать бесплатную версии DBeaver через Интернет и следовать инструкциям с сайта.

Говоря о функциональных возможностях, отметим, что при помощи DBeaver выполняются все основные действия с базой данных; для вызова операций имеется хорошо сгруппированное меню, кнопки на панели инструментов, а наиболее важные функции вызываются при помощи горячих клавиш. Основные компоненты DBeaver:

- диспетчер соединений позволяет установить соединение с базой данных, выбрав требуемый драйвер. Если соединений много, то их можно организовать в папки, что особенно удобно, когда в одной папке сгруппированы соединения, используемые одним приложением;
- браузер метаданных, который показывает соединения и их содержимое, с его помощью можно просмотреть существующие таблицы, представления, столбцы, индексы, процедуры, триггеры, настройки безопасности (пользователи, роли, и т.д.);
- редактор SQL, в котором можно создавать (или импортировать готовые) скрипты, редактировать SQL запросы, создавать шаблоны и т.п.;
- редактор представлений, позволяющий выбрать способ представления данных (JSON, plain-text, XML), прокрутку большого текста, настроить работу с типами данных BLOB/CLOB (режим просмотра и редактирования), фильтрацию данных и т.п.;
- поиск данных/метаданных осуществляет полнотекстовый поиск или по заранее заданным условиям;
- сравнение структуры баз данных (сравнению подлежат только однотипные объекты: таблицы, схемы, базы данных целиком);
- импорт/экспорт баз данных. Поддерживаемые форматы файлов: CSV, HTML, XML;

- ER-диаграммы (создаются автоматически), диаграмма может быть экспортирована в файлы следующих форматов: GIF, PNG, BMP, GraphML. Заметим, что диаграмма может быть создана и для NoSQL баз данных, при этом коллекция будет представлена как таблица, а документ — как строка таблицы;
- менеджер запросов, который позволяет сохранять все ранее выполненные запросы и статистику их выполнения (время выполнения, количество принесенных / обновленных строк, ошибки и т.д.).

Также имеется возможность администрирования пользователей (создать, удалить, изменить данные пользователя, назначить ему привилегии), посмотреть системную информацию и произвести настройки.

Кроме того, для различных баз данных в DBeaver имеются специфические возможности. Например, для такой распространенной базы данных, как MySQL имеются возможности управления сеансами, пользователями, каталогами и пр.

Еще одним важным свойством является возможность редактирования содержимого ячеек, для чего достаточно выбрать нужную таблицу реляционной СУБД или коллекцию СУБД NoSQL. При этом в случае выбора СУБД NoSQL документ коллекции будет представлен в виде таблицы.

Таким образом, универсальный бесплатный менеджер баз данных DBeaver может с успехом использоваться разработчиками и администраторами баз данных, а также при обучении студентов ИТ-направлений приемам работы с SQL и NoSQL базами данных [2].

Литература

[1] DBeaver [Электронный ресурс]. Режим доступа: `http://dbeaver.jkiss.org/`, свободный. — Загл. с экрана. — Яз. англ. Дата обращения: 15.01.2016.

[2] Мартишин С.А., Симонов В.Л., Храпченко М.В. *Разработка информационных систем с использованием СПО NoSQL СУБД MongoDB* // Сб. тезисов десятой конференции «Свободное программное обеспечение в высшей школе», НОУ «ИПС-Университет г. Переславля им. А.К. Айламазяна», г. Переславль-Залесский, 24-25 января 2015, стр. 72-75.

Евгений Кондратьев
Москва, Институт высоких технологий Московского технологического университета

Элементы численных методов в LibreOffice Calc

Аннотация

В LibreOffice Calc реализованы элементы численных методов. Даются примеры использования разработанных средств.

При подготовке в высшей школе кадров по многим направлениям для промышленного производства обучение численным методам на персональном компьютере (ПК) традиционно в настоящее время даётся в Mathcad или в MATLAB. Эти программы требуют серьёзной подготовки для работы с ними и являются коммерческими продуктами. Также для этих целей часто используют программирование. Но этот путь для многих численных методов на ПК требует хорошего владения навыками и опыта программирования.

Электронная таблица Calc, входящая в состав свободного программного обеспечения (СПО) LibreOffice The Document Foundation [1], позволяет доступно обычному пользователю ПК использовать многие численные методы при решении различных инженерных задач. Переход на СПО в федеральных бюджетных учреждениях определен планом, утвержденным распоряжением Правительства РФ от 17 декабря 2010 г. № 2299-р [2].

В LibreOffice Calc реализованы: численное интегрирование; основные одношаговые методы численного решение задачи Коши для обыкновенных дифференциальных уравнений первого порядка, системы обыкновенных дифференциальных уравнений первого порядка и обыкновенных дифференциальных уравнений высших порядков; многошаговые методы Адамса-Башфорта решения задачи Коши для обыкновенных дифференциальных уравнений первого и второго порядков.

На рис. 1 приведен пример [3] решения в Calc двухшаговым методом Адамса-Башфорта следующей задачи Коши:

$$\begin{cases} y' = 2xy \\ y(0) = 1 \end{cases} \tag{22.1}$$

на отрезке [0, 1] с шагом $h = 0.05$. Точным решением этой задачи является функция $y = exp(x^2)$. Это средство по простоте похоже на метод Эйлера в Calc [4], но намного точнее его.

На рис. 2 приведены графики решения в Calc методом Рунге-Кутты четвертого порядка [5] и четырёхшаговым методом Адамса-Башфорта задачи Коши для обыкновенного дифференциального уравнения второго порядка:

$$\begin{cases} y'' = x^2 + y^2 \\ y(0) = 0 \\ y'(0) = 1 \end{cases} \tag{22.2}$$

на отрезке [0, 2] с шагом $h = 0.05$, для которого нельзя явно получить точного решения.

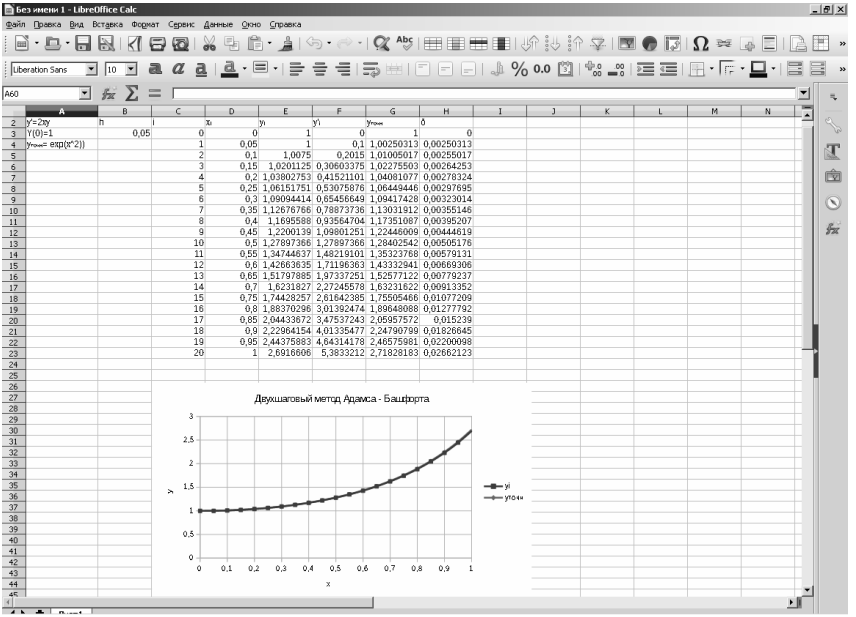


Рис. 1: Решение в Calc двухшаговым методом Адамса-Башфорта задачи Коши (22.1).

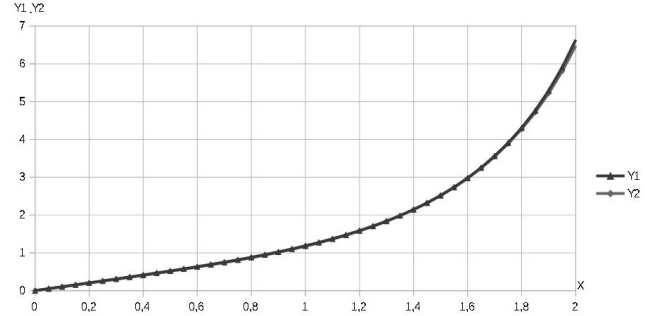


Рис. 2: Графики решения в Calc методом Рунге-Кутты 4-го порядка (кривая Y1) и четырёхшаговым методом Адамса-Башфорта (кривая Y2) задачи Коши (22.2).

Разработанные средства доступны обычному пользователю ПК, и часть из них внедрена в учебный процесс [4, 6].

Литература

[1] `http://ru.libreoffice.org/`

[2] *Об утверждении плана перехода федеральных органов исполнительной власти и федеральных бюджетных учреждений на использование свободного программного обеспечения на 2011–2015 г.г.*: распоряжение Правительства РФ от 17 декабря 2010 г. №2299-р., `http://base.garant.ru/6746035/`

[3] Кондратьев Е. М., *Двухшаговый метод Адамса-Башфорта в Calc* // The Way of Science., 2015., № 11 (21)., с. 52–57.

[4] Кондратьев Е. М., *Метод Эйлера в Calc*: методические указания., М.: Изд-во МГУПИ, 2013., 16 с.

[5] Кондратьев Е. М., *Численное решение в Calc задачи Коши для обыкновенных дифференциальных уравнений высших порядков* // The Way of Science., 2015., № 2 (12)., С. 30–35.

[6] Кондратьев Е. М. *Численное интегрирование в Calc*: методические указания., М.: Изд-во МГУПИ, 2014., 24 с.

Илья Захаров
Москва, SoftAccord

Метаданные языков визуализации, специфицирования, конструирования и документирования (языки ВСКД) на примере UML/SysML

Аннотация

Создатели UML в первых строках Usere Guide[6] называют его языком визуализации, специфицирования, конструирования и документирования. UML — на сегодняшний день самый распространенный из таких языков, кратко — языков ВСКД. Их основная особенность — непосредственная взаимосвязь с современными естествознанием и философией. Отсюда возникают некоторые проблемы с изучением и использованием языков ВСКД, сначала привычно трактуемых с точки зрения вопросов программирования и разработки информационных систем, — при том, что вся грамматика UML-2 составляет всего лишь полсотни обозначений. Для понимания и использования языков ВСКД необходимо назвать и рассмотреть их метаданные: прежде всего — онтологический смысл понятия «объект» в современной теории познания, а также категориальные сетки и существующие в их рамках наборы понятий, посредством которых можно «понимать» объекты, проектировать и анализировать их, описывать их характеристики. Кроме того, у любой разработки объективно существуют «предельные основания» — исходные посылки, цели, которые не могут быть рационально обоснованы, т. е. предмет веры. В качестве такой «первопричины» разработки UML сами правообладатели называют идею MDA. Доклад посвящен, прежде всего, вербализации вышеназванных вопросов — действительно, в большинстве текстов по UML они даже не упоминаются. Сделаны попытки начать обсуждение этих важнейших вопросов.

Начать изложение хотелось бы с интересного вывода из книги «Теоретическое знание»[3], который можно считать обоснованием предлагаемых в докладе сопоставлений: «...фундаментальные теории не являются продуктом индуктивного обобщения опыта, а создаются вначале за счет трансляции концептуальных средств, заимствованных из других областей теоретического знания, и только затем обосновываются опытом...».

Из истории языков ВСКД известно, следующее: «когда все согласились, что модульные компьютерные программы лучше монолитных, начался спор о том, как строить модули ... [остановились на концепции «объекта»] ... сделать так, чтобы каждый модуль соответствовал одной и только одной вещи в реальном мире»[5].

Понятие «объект» в онтологии научного метода познания

Для раскрытия онтологического смысла понятия «объект» обратимся к общепризнанным работам[3] по теории познания. Вот несколько тезисов в сокращенном изложении.

«...если исчезнут инвариантные признаки науки, отличающие ее от других форм познания (искусства, обыденного познания, философии, религиозного постижения мира), то это будет означать исчезновение науки.»

«Наука ставит своей конечной целью предвидеть процесс преобразования предметов практической деятельности (объект в исходном состоянии) в соответствующие продукты (объект в конечном состоянии). Это преобразование всегда определено сущностными связями, законами изменения и развития объектов...»

«Поскольку в деятельности могут преобразовываться самые различные объекты — предметы природы, человек (и состояния его сознания), подсистемы общества, знаковые объекты, функционирующие в качестве феноменов культуры и т.д., — постольку все они могут стать предметами научного исследования.

Ориентация науки на изучение объектов, которые могут быть включены в деятельность (либо актуально, либо потенциально как возможные объекты ее будущего преобразования), и их исследование как подчиняющихся объективным законам функционирования и развития составляет первую главную особенность научного познания.

Эта особенность отличает его от других форм познавательной деятельности человека.»

«Наука в человеческой деятельности выделяет только ее предметную структуру и все рассматривает сквозь призму этой структуры. Как царь Мидас из известной древней легенды — к чему бы он ни прикасался, все обращалось в золото, — так и наука, к чему бы она ни прикоснулась, — все для нее предмет, который живет, функционирует и развивается по объективным законам.»

Таким образом, концептуальная ориентация на понятие «объект» в языках ВСКД жёстко и однозначно относит их к инструментам именно научной деятельности, которая в свою очередь существует в следующем пространстве форм познавательной деятельности:

- наука;
- обыденное познание;
- философия;
- религия (если говорить о методе, то лучше было бы сказать — теология, ещё точнее — апофатическая теология, так как всё остальное в основном относится к сфере религиозной философии);
- искусство.

Фундаментальным свойством научного метода является представление обо всём как об «объекте», включая «объективизацию» других форм познания, что можно интерпретировать как универсальный метод их подмены, которая в некоторых случаях равносильна их уничтожению.

Так как для науки всё есть объект, становится достаточно ясной весьма невнятно излагаемая в книгах по языку UML тема «классификаторов» и неожиданные интерпретации почти любой конструкции языка как объекта.

Категориальные сетки и пространства понятий, характеризующих объекты различной сложности

Грамматики языков ВСКД представляют собой наборы взаимосвязанных понятий, посредством которых и представляется возможность осуществлять визуализацию, специфицирование, конструирование и документирование объектов.

Для понимания этих грамматик целесообразно сравнить их с современными представлениями об объектах в естествознании.

В западной литературе, традиционно восходящей к работам Людвига фон Берталанфи и Ильи Пригожина, важнейшей категорией, характеризующей объект, является «сложность», что можно интерпретировать как одномерную категориальную сетку, в рамках которой в хронологическом порядке отражается возникновение новых понятий, раскрывающих смысл темы «сложности».

В статье Ратникова В.С. «Обновление методологической культуры в процессе освоения наукой феномена сложности», опубликованной в рамках сборника статей[2], можно проследить хронологию нарастания инструментария работы науки с категорией «сложность».

Начиная с классической науки, её можно представить в виде такой последовательности:

I этап мир прост в смысле его фундаментального устройства, сложность связана с недостатком информации — — **II этап** применение средних значений характеристик состояния (Клаузиус) — — применение вероятностных значений распределения величин (Максвелл) — — статистические закономерности на фоне индивидуального движения отдельных частиц, эргодичность (Больцман) — — вероятностные характеристики единичного микрообъекта (квантовая механика) — — системно-структурный подход, т. е. рассмотрение числа элементов, их связей, в т.ч. обратных связей, степени и разнообразия взаимодействия (кибернетика) — — стохастичекие системы — — концепция информации и управления — — соотношение сложности, организованности и разнообразия, уровни организации системы — — понятие иерархической системы — — модели массовой и индивидуальной случайности, внутренней сложности (Бернштейн) — — живые системы, эволюционирующие системы — — **III этап** отказ от приоритета простоты, как методологического принципа научного познания — — простые объекты со сложным поведением, т. е. «сложность простоты» — — нелинейная динамика — — самоорганизация как свойство сложных систем — — эволюция как направление самоорганизации — — неустойчивость и неравновесность, соотношение спонтанности и детерминистичности, роль флуктуаций — — взаимодополнение жёсткодетерменистских и вероятностно-статистических моделей — — взаимопереходы порядка и хаоса, фрактальные аттракторы, модели с наложением внутренней сложности на несколько ключевых параметров — — существование пределов предсказуемости поведения динамических систем.

Категориальную сетку, предложенную В. С. Стёпиным[3], и распределение в ней понятий, характеризующих сложный объект, можно представить в форме таблицы.

Таблица 1: Категориальная сетка (В. С. Стёпин)

Категориальная сетка сложности объекта	**Механические (простые) системы**	**Сложные саморегулируемые системы**	**Сложные саморазвивающиеся системы**
	$10^1 - 10^3$ элементов	$10^3 - 10^6$ элементов; примеры: завод-автомат, биологические системы в аспекте их функционирования	$10^6 - 10^{16}$ элементов, способны увеличивать свою сложность
Часть-целое	Свойство целого определяется свойствами частей, часть может быть выведена из целого.	Целое — главное. Есть системные свойства. Свойства частей частично определяют свойство целого.	Целое не сводится к частям, управляет ими, изменяет их, возникают новые функции целого
Вещь-процесс	Вещь первична. Процесс — результат взаимодействия между вещами.	Процесс первичен. Вещь — это устойчивое состояние процесса.	Вещь — воспроизводящийся процесс (как правило, структура остается, а наполнение меняется). Качественное изменение вещи, переход от одного гомеостаза к другому.
Причинность	Лапласовский детерменизм (Л.д.): если одновременно известны условия и причины, то можно предсказать поведение с любой точностью на любов промежутке времени. Стохастические процессы могут быть сведены к Л.д., их наличие — лишь признак незнания.	Лапласовская + вероятностная причинность; обратные связи, стохастическое взаимодействие, которые первичны. Случайные события в подсистемах. Существует целевая причинность.	Лапласовская + вероятностная + целевая. (Странный аттрактор: случайный выбор и ослабевание других факторов — квазицель, фазовый переход.) «Кольцевая причинность»: следствие становится причиной для новых явлений.
Пространство-время	Время и пространство внешние, то есть не влияют на процессы. Интервалы при переходе между системами отсчета не изменяются.	Существует внутренне пространство-время. На одном физическом пространстве существуют разные экологические ниши.	Внутренне время, переходы от одного к другому времени.

Универсальные паттерны для моделирования любых объектов

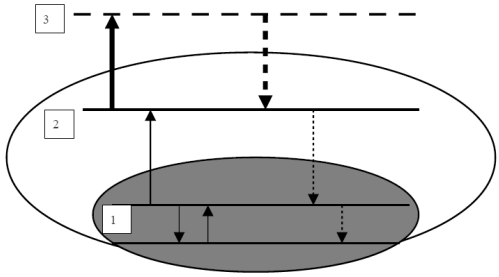
Анализ категориальных сеток и систем понятий, характеризующих объекты различной сложности, указывает на существование универсальных паттернов, парадигмальных моделей.

Этот факт абсолютно подтверждается историей естествознания и является ценным прикладным результатом, так как на практике оказывается невозможным выйти за пределы существующих паттернов никому из учёных соответствующей эпохи.

Для 1-го этапа (см. Ратников) и 1-го столбца таблицы (см. Стёпин) таким «паттерном» является устройство механических часов. Этот факт обозначен в большом количестве самых разных текстов и широко обсуждается.

Для условно 2-го этапа (см. Ратников) и 2-го столбца таблицы (см. Стёпин) «паттерном» является самоорганизующийся автомат с гомеостазом: блок управления и блок информации с системными параметрами; обратные связи, стохастические процессы в подсистемах. Этот факт явно был назван В.С. Стёпиным, его широкое обсуждение не выглядит заметным.

Для условно 3-го этапа (см. Ратников) и 3-го столбца таблицы (см. Стёпин) «паттерном» является предложенная В.С. Стёпиным[1] схема возникновения новых уровней саморегуляции у открытых систем с саморазвитием.

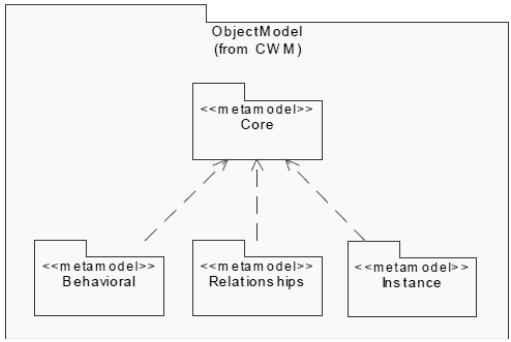


1. Исходная саморегуляция.
2. Новый тип саморегуляции, основанный на трансформации предшествующих уровней иерархии системы.

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

Это перспективная парадигмальная модель, работу с которой ещё предстоит освоить.

Вместе с тем, в настоящее время в иностранных источниках, особенно метамоделях, разработанных на UNL/SysML часто встречается «паттерн» типа «ядро-оболочка». Вот пример из документа «Common Warehouse Metamodel (CWM) Specification».



Описание ещё одной парадигмальной модели можно увидеть в вышеназванной статье Ратникова, хотя явно она не названа. Речь идёт о модели «наложения» внутренней сложности на несколько ключевых параметров.

Такой «паттерн» можно представить диаграммой компонентов в UML, когда «сложность» подразумевается внутри небольшого количества «черных» ящиков, а сами компоненты — «чёрные ящики» вместе с их связями и портами образуют простую обозримую понятную схему. Этот «паттерн» можно было бы назвать, например, «внутренняя сложность».

Итого в распоряжении современных ученых есть пять парадигмальных моделей. Их наличие и содержание даёт обоснованный ответ на вопрос, часто обсуждаемый в книгах по UML: в какой последовательности надо составлять диаграммы.

Предельные основания разработки UML/SysML

Воспользуемся цитатами из книги[4] авторов, близких к первоисточникам.

«MDA [«Архитектура, управляемая моделью»] — не процесс разработки. Это не спецификация. Это не реализация. Это не комплекс мер по согласованию. MDA не имеет эталонной реализации».

«Инструменты MDA преобразуют «чистые» бизнес-модели в полноценные, развёртываемые, исполняемые приложениями с минимумом решений технологического плана. Модификация «чистой» бизнес-модели, заложенной в основу приложения, требует только обновлений в зависимых технологических областях... поэтому приложение удаётся заново сгенерировать за несколько минут».

«Хотя инструментарий MDA в конечном итоге надеется заместить все языки программирования, в своём текущем состоянии он не обладает достаточной изощрённостью... »

«Цель MDA просто формулируются, но до её полноценной реализации ещё далеко».

То есть MDA направлено в идеале на создание универсальных объектов — «кнопок», нажатие на которые приводит, например, к распечатке на 3D-принтере нужной вещи или изменению порядка вещей.

Мне кажется, что предыдущие пятьсот лет эта концепция называлась...

Выводы

Выявление и изучение метаданных языков ВСКД способствует их изучению, пониманию и использованию.

Без них невозможно понимать перспективы и пути дальнейшего развития как самих языком ВСКД, так и методов, с ними связанных.

Кроме того, изучение метаданных позволяет получить обоснованные ответы на «трудные» вопросы, слабо освещенные в литературе по языкам ВСКД.

Литература

[1] *Киященко Л. П., Степин В. С., коллектив авторов*, Постнеклассика. Философия, наука, культура. — СПб.: Міръ, 2009. — 672с., ISBN 978-5-98846-037-4

[2] *Лисеев И. К., Садовский В. Н., коллектив авторов*, Системный подход в современной науке. — М.: Прогресс-Традиция, 2004. — 563с., ISBN 5-89826-146-X

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

[4] *Dan Pilone, Neil Pitman*, UML 2 для программистов. — СПб.: Питер, 2012. — 240с., ISBN 978-5-459-01684-0

[5] *Edward Yourdon, Carl Argila*, Структурные модели в объектно ориентированном анализе и проектировании. — М.: ЛОРИ, 1999. — 268с., ISBN 5-85582-057-2

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

Евгений Алексеев, Пётр Демин
Киров, ФГБОУ ВО «Вятский Государственный Университет»
http://evgeniyalekseev.wordpress.com

Свободные библиотеки в вычислительных задачах

Аннотация

Представлен опыт совместного использования компиляторов g++, gfortran и свободных математических библиотек для решения вычислительных задач. Особое внимание уделено решению вычислительных задач высокой точности. Рассмотрены возможности библиотек mpfr и c-xsc. Также представлены программные средства, позволяющие использовать графические возможности gnuplot.

При решении вычислительных задач основными языками программирования, позволяющими разрабатывать быстродействующие приложения являются C(C++) и Фортран. Кроме стандартных средств языков программирования, в этих языках реализованы современные технологии параллельного программирования. Это позволяет использовать C и Фортран не только для решения задач на локальном персональном компьютере, но и при реализации задач на вычислительных кластерах.

В современных свободных компиляторах (gcc, g++, gfortran) реализованы все возможности современных языков C и Фортран. Компиляторы gcc, gfortran используются на всех современных кластерах

наряду с компиляторами Intel. При решении вычислительных задач пользователь может столкнуться с несколькими проблемами:

- разработка и реализация высокоэффективных алгоритмов вычислительной математики;
- достижение достаточной точности вычислений;
- кросс-платформная реализация графического вывода результатов.

Имеет смысл использовать свободную научную библиотеку языка C GSL[1] при реализации алгоритмов вычислительной математики. Библиотека позволяет решать:

- некоторые задачи линейной алгебры (арифметические операции с матрицами (векторами) и поэлементное произведение, LU- и QR-разложение, нахождение собственных чисел и векторов);
- задачи локальной оптимизации (метод золотого сечения, метод Брента одномерной оптимизации и градиентные методы многомерной оптимизации Флетчера-Ривса, Полак-Райбьера и др.);
- задачи статистики (генерация и работа со случайными и квазислучайными последовательностями, нормальное распределение, распределение Пуассона и др);
- задачи численного интегрирования (в основе метод Гаусса-Кронрода с различным количеством точек, есть алгоритм с исключением особых точек, функция вычисления интеграла для колебательных функций, функция вычисления интеграла Фурье и др.);
- задачи с комплексными числами(арифметические операции, модуль, аргумент, поиск сопряженного, обратного, противоположного по знаку, возведение в степень, вычисление тригонометрических функций от комплексного числа и др);
- производить сортировку(пирамидальная сортировка) элементов массива (вектора).

Библиотека GSL предоставляет свой интерфейс для доступа к базовой библиотеке векторных и матричных операций blas.

Существует интерфейс для использования библиотеки при написании программ на Фортране[2].

[1] GNU Scientific Library — http://www.gnu.org/software/gsl/

[2] `http://www.lrz.de/services/software/mathematik/gsl/fortran/`

Довольно сложной проблемой при решении инженерных и математических задач является проблема точности вычислений. Встроенные в язык типы данных не позволяют гарантировать правильность вычислений при арифметических операциях с числами разных порядков.

На сегодняшний день существует несколько подходов для решения этой проблемы. Авторами рассмотрены два подхода:

- использование интервального анализа для вычислений, существует несколько библиотек на C(C++) для реализации интервального анализа [1,2], одной из них является C-XSC[3]. Библиотека имеет открытый исходный код. Имеет возможность работы с комплексными числами, матрицами и векторами. Также имеет ряд типов данных с префиксом «$l\_$», реализующих вычисления с повышенной точностью в интервалах. Имеется возможность распараллеливания некоторых функций с помощью OpenMP.

- использование вещественных чисел произвольной точности — (MPFR[4]) — библиотеки языков C и C++, основанные на GMP. Поддерживает стандарт IEEE-754 для арифметических операций с числами с плавающей точкой. Позволяет инициализировать переменные с большой точностью, производить с ними арифметические операции, возводить их в степень, находить корень, вычислять логарифм и экспоненту, а также тригонометрические функции и константы. За счет возможности выбора количества бит, отводимое на хранение переменной, можно регулировать точность вычислений.

Одной из проблем при написании вычислительных консольных приложений на C(C++) и Фортране является вывод графиков с результатами работы программы. Для решения этой проблемы предлагается использовать возможности программы GNUPlot. Для этого необходим интерфейс между gfortran, g++, gcc и GNUPlot. В качестве интерфейса между Фортраном и GNUPlot можно предложить[5] gnufor2, а между C и Gnuplot [6] — gnuplot-cpp.

[3] `http://www.xsc.de`

[4] `http://www.mpfr.org`, `http://www.holoborodko.com/pavel/mpfr`

[5] `http://www.math.yorku.ca/~akuznets/gnufor2/`

[6] `https://code.google.com/p/gnuplot-cpp`

Литература

[1] *Кулиш У., Рац Д., Хаммер Р., Хокс М.* Достоверные вычисления. Базовые численные методы. — Москва-Ижевск: РХД, 2005. — 496с.

[2] *Шарый С. П.* Конечномерный интервальный анализ. — URL: `http://www.nsc.ru/interval/Library/InteBooks/SharyBook.pdf`.

Карпеш С. В., Амелькин С. А., Клементьев А. Д.
Переславль-Залесский, ЧОУ ВО «УГП им А.К. Айламазяна»

Разработка аппаратного обеспечения для системы управления высокопроизводительным вычислительным комплексом с погружным охлаждением

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. Данный доклад посвящен процессу и особенностям разработки контроллеров для системы управления погружным охлаждением высокопроизводительного вычислительного комплекса.

Для контроля теплового режима вычислительных узлов и поддержания температуры охлаждающей жидкости, не превышающей заданную, необходимы средства управления. Их реализация требует алгоритм, учитывающий все особенности системы. Входными параметрами для алгоритма являются показания с датчиков температуры, а выходные данными являются управляющие сигналы для насоса и вентиляторов.

Для реализации программной части используются SCADA системы. Это комплект ПО, предназначенный для разработки и обеспечения работы в реальном времени систем сбора, обработки и архивирования информации об объекте мониторинга. В проекте выбрана

система OpenSCADA. Это свободная модульная система, разрабатываемая энтузиастами, однако, несмотря на этот факт, по стабильности не уступающая коммерческим аналогам.

Вычислительный комплекс состоит из двух блоков: бак с вычислительным оборудованием и блок охлаждения.

В баке установлены нескольких узлов, коммутационное оборудование и управляющий компьютер. Ролью последнего является контроль теплового режима всей системы.

Так как нагрузка между узлами может быть распределена не равномерно, то одним датчиком температуры для бака не обойтись. Температуру необходимо измерять у каждого процессора, которых в каждом модуле может быть несколько. Обычно это число равно двум, но стоит заранее предусмотреть возможность установки большего количества датчиков.

Из существующих контроллеров весьма проблематично подобрать компактное решение. Количество входов в них меньше, чем требуется. Самым распространенным числом является 4. Используются аналоговые термометры, предусматривающие подведение к каждому из них отдельных проводников, что значительно усложняет монтаж.

Аналоговые датчики имеют ещё один значительный минус. Для них требуется калибровка и защита от наводок. При размещении контроллера за пределами бака длина проводников сильно возрастает и на переходной плате требуются дополнительные разъемы, что уменьшает надежность. Размещая контроллеры в баке, тратится ценное место.

Решением является создание собственного контроллера с использованием цифровых термометров.

Из их плюсов можно отметить:

– Возможность установки нескольких термометров на один «луч»;
– Калибровка изготовителем на заводе;
– Возможность установки на большем расстоянии от контроллера с минимальными потерями в точности измерений.

Для установки внутри бака были выбраны датчики на шине I^2C в связке с мультиплексором. Основным их преимуществом является адресация. Структура блока с вычислительным оборудованием подразумевает на каждый узел луч с несколькими датчиками. При использовании нескольких датчиков 1-Wire на одной шине необходимо обращаться по идентификатору, который для каждой микросхе-

мы уникален. Это подразумевает перепрограммирование контроллера при любом изменении порядка или замене вышедшего из строя датчика.

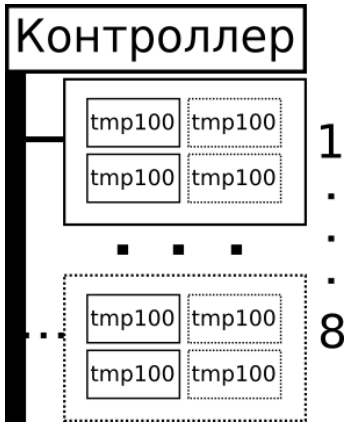


Рис. 1: Принципиальная схема бака с вычислительными узлами

Считывание данных с датчиков не является сложной задачей, в отличии от процесса общения между контроллером и OpenSCADA. Модуль сбора данных запрашивает данные с контроллера по протоколу Modbus. Это открытый и подробно документированный протокол, основанный на архитектуре ведущий-ведомый. Данные передаются пакетами. Посылка содержит в себе идентификатор устройства, команду, данные, необходимые для выполнения команды, и контрольную сумму. Ответ содержит идентификатор ведомого устройства, команду, результат работы и контрольную сумму.

Обязательным элементом устройств подобного класса является сторожевой таймер, при переполнении которого происходит перезагрузка микросхемы. Этот механизм позволяет автоматически восстановить работу устройства при зависании. Время срабатывания подбирается исходя из самой долгой операции при работе. Для контроллера самым долгим промежутком будет отправка даных. Размер пакета ограничен 252 байтами с данными. Плюс 4 байта служебной информации. На скорости 9600 бод они будут передаваться за ≈ 0.27 сек.

Во время работ над проектом написана библиотека для работы с Modbus RTU для микроконтроллеров AVR, поддерживающая все основные функции чтения и записи.

В спецификации [1] их описано 8, но условно можно разделить на 3 типа: чтение, запись и множественная запись. Различаются они только по типам данных, с которыми работают.

Библиотека основана на прерывании по приему байта. Под этим подразумевается не только сохранение пакета с данными, но и мгновенная обработка запроса, после окончания его приема. Такое решение позволяет ввести некоторую многозадачность при работе микроконтроллера и «заниматься своими делами» в основном цикле программы, но при этом не беспокоиться о том, что запрос может быть пропущен.

Объем памяти, занимаемый ядром библиотеки, составляет 20 байт. Остальная память микроконтроллера доступна для хранения регистров. Однако необходимо учитывать тот факт, что максимальная длина пакета составляет 256 байт, которые должны быть доступны при получении запроса. На генерацию ответа выделение памяти не требуется, а контрольные суммы считаются по ходу передачи. Использован табличный метод подсчета CRC, данные для которого хранятся во flash памяти.

Литература

[1] Спецификации протокола MODBUS `http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf`

Дуванов Александр Александрович, Первин Юрий Абрамович
Переславль-Залесский, НОУ "Роботландия Университет города Переславля имени А.К.Айламазяна
http://РОБОТЛАНДИЯ.рф

Роботы, алгоритмы и программы в дистанционном курсе раннего обучения информатике «Азбука Роботландии»

Образовательное учреждение дополнительного образования детей и педагогов «Роботландия+», выросшее из лаборатории школьной информатики Института программных систем РАН, работает в области информатизации школьного образования с начала 90-годов XX века. Программная и методическая продукция учреждения известна в стране и за рубежом ([1], [10]), она стала основой первого в стране непрерывного школьного курса информатики «Информационная культура» ([7]). При построении концепций системы ориентиром её проектирования стали идеи сибирской школы программирования академика А.П. Ершова [6],[10]. Её главные объекты учебной деятельности — исполнители, программные модели роботов. Из обоснований педагогических целей непрерывного курса школьной информатики однозначно вытекало одно из следствий этого обоснования — раннее обучение информатике [6, 8].

Поначалу экспериментальным полем были занятия дополнительного образования в переславских школах, а также регулярные летние лагеря-школы юных программистов на берегу Плещеева озера. Однако скоро самой значимой формой деятельности стали программистские соревнования по программированию — олимпиады школьников, организуемые средствами электронной почты ([3]). И как только удалось технологически преодолеть барьер расстояния, встал вопрос о серии курсов дистанционного обучения (ДО), объединяемых в дистанционный центр (ДЦ) «Роботландский сетевой университет»[1] ([4], [5], [2], [9]).

[1] http://РОБОТЛАНДИЯ.рф

Практика этого центра ДО — лучший аргумент в пользу раннего обучения информатике и против скептиков привлечения младших школьников к дистанционным формам обучения .

К терминологии ДО:

коллективный ученик, **команда** — коллектив детей, руководимый учителем и участвующий в дистанционном учебном процессе как единый адресат; учитель собирает в команду детей одного возраста;

курс — подразделение ДЦ, группирующее преподавателей и слушателей по тематическому принципу;

куратор — руководитель курса; обычно, куратор — разработчик одного (или нескольких) компонентов дистанционного учебного процесса — автор учебника для школьников или пособия для учителя, программист-разработчик программного продукта;

тема — часть курса, ограниченная временными рамками и группирующая учебный материал по признакам содержания; курс содержит несколько тем; каждая тема, в свою очередь, реализуется несколькими занятиями.

занятие — аудиторный урок команды с учителем — руководителем команды — в рамках базовой формы учебного процесса или факультатива по методическим материалам, получаемым из ДЦ;

конкурс — форма учебного процесса, группирующая несколько занятий в теме и позволяющая внести в занятия элементы соревнования, стимулирующие учащихся; конкурсы позволяют привести неформальные показатели контроля результатов учебного процесса к количественным характеристикам; конкурсы могут быть **командными** и **личными**;

перекрёстная проверка — способ проверки конкурсных заданий, в которой наряду с методистами ДЦ принимают участие слушатели — учителя и руководимые ими команды;

система оценивания ученических работ (**экспертная система**) — многокритериальная система оценок, при которой отдельные качества работы оцениваются в баллах.

Проверяя работы далеких сверстников из других команд, ученик самоутверждается, сравнивая свои результаты с тем, что получили его товарищи (воспитательный компонент воздействия методической системы на ученика) или обнаруживает свои погрешности, исправление которых является частью ключевой компетенции — **научиться учиться.**

Демонстрируются фрагменты уроков из курса «Азбука Роботландии».

Литература

[1] *Гольцман М. А., Дуванов А. А., Первин Ю. А.* Концепции развития ребенка в курсе «Роботландия». — В кн. «Материалы советско-французского семинара СИНТЕЗ по компьютеризации начального образования», тетрадь 1, доклады советских участников, Переславль-Залесский, М., 1992. — сс. 7–14.

[2] *Дуванов А. А., Первин Ю. А.* Необыкновенные приключения Пети Кука в Роботландии. — Финансы и статистика. 2-е изд., — М., 273 с.

[3] *Дуванов А. А., Первин Ю. А. Скородумов В. С.* Дистанционные формы раннего информатического образования. — В кн. Материалы конференции ИНО 97 «Информатизация непрерывного образования», М., 1997. сс. 39–41.

[4] *Дуванов А. А., Рудь А. В., Семенко В. П.* Азы программирования. Факультативный курс. Книга для учителя (496 с.). Книга для ученика (352 с.). Задачник (160 с.). — С.-Пб. — 2005.

[5] *Дуванов А. А., Шумилина Н. Д.* «Азбука Роботландии»: курс информатики для малышей в алгоритмическом изложении. — Вестник Ярославского регионального отделения РАЕН — т. 8, № 1, 2014. — сс. 40–51.

[6] *Ершов А. П., Звенигородский Г. А., Первин Ю. А.* Школьная информатика (концепции, состояние, перспективы). — Препринт ВЦ СО АН СССР, № 152, Новосибирск, 1979. — 32 с.

[7] *Коган Е. Я., Первин Ю. А.* Курс «Информационная культура» — региональный компонент школьного образования. — «Информатика и образование», № 1, 1995. — сс. 3–14.

[8] *Первин Ю. А.* Методика раннего обучения информатике. — М., «Бином», Лаборатория базовых знаний, 2-е издание, 2008. — 288 с.

[9] *Первин Ю. А.* Информатика дома и в школе. Основы информатики. Книга для учителя (144 с.). Книга для ученика (352 с.) — БХВ. — С-Пб. — 2003.

[10] *Первин Ю. А.* Школьная информатика. Сегодняшние уроки наследия А.П.Ершова — III Международная конференция памяти академика А.П.Ершова «Перспективы систем информатики PSI 99». Школьная информатика». — Доклады и тезисы. Новосибирск, 1999. — сс. 45–53.

А. А. Демидов
Переславль-Залесский, ИПС им. А.К.Айламазяна РАН*

Введение в алгебраические вычисления: две стадии работы алгоритма

Аннотация

В настоящее время с целью создания экзафлопсных супер-ЭВМ, способных выполнять 10^{18} операций с плавающей точкой в секунду, идёт интенсивный поиск новых решений и идей, позволяющих преодолеть известные ограничения производительности, такие как предел Ландауэра [1]. В работе рассматривается алгебраический подход к вычислениям, который позволяет устранить эффект «стены памяти». В частности, устанавливается, что выполнение алгебраического алгоритма состоит из двух этапов, имеющих различную вычислительную сложность: во-первых, настройка структуры вычислительного автомата, и во-вторых, получение решения путём распространения входного сигнала. Наличие первого этапа до сих пор не оговаривалось явно, также не ставилась задача исследования временных издержек этого этапа вычисления.

Введение

Алгебраический подход к вычислениям возник в начале 1960-х годов на стыке теории алгоритмов и теории автоматов в связи с необходимостью внедрения блочного подхода к синтезу схем стремительно усложнявшихся электронно-вычислительных устройств [2, 3]. Каждому конечному автомату можно поставить в соответствие элемент некоторой полугруппы так, что композиции автоматов будет соответствовать композиция отображений, при этом выходные сигналы одного автомата используются как входные сигналы другого [4]. С другой стороны, каждому завершающемуся алгоритму можно поставить в соответствие своё отображение — тоже элемент полугруппы. Это позволяет установить соответствие между завершающимися алгоритмами и реализующими их конечными автоматами.

[1]Институт программных систем им. А. К. Айламазяна РАН, Исследовательский центр искусственного интеллекта, ул. Петра I, д.4а, с. Веськово, 152021, Переславский р-н, Ярославская обл.; м.н.с., e-mail: alex@dem.botik.ru

Новое развитие алгебраический подход получает в настоящее время в связи с работами Н. Н. Непейводы [5, 6], предложившим аксиоматику программных алгебр GAPS (Generic algebraic program structure). Принципиальное отличие новой аксиоматики состоит в том, что в дополнение к ассоциативной операции «$\circ$» композиции программ GAPS содержат дополнительную неассоциативную операцию «$*$» аппликации (применения) программы к данным; обе операции связаны тождествами

$$\begin{gathered}(x * f) * g = x * (f \circ g);\\ 0 * x = x * 0 = 0;\\ x * e = x.\end{gathered} \tag{27.1}$$

Необходимость неассоциативной операции можно продемонстрировать на следующем примере.

Пример 1. Пусть имеется программа p, принимающая на вход целое число n и посылающая на выход число $n + 1$. Понимая p как отображение, можно записать $n + 1 = p(n)$; двукратное применение даст $n + 2 = p(p(n))$, чему формально соответствует суперпозиция $n + 2 = [p \circ p](n)$: «применить программу к самой себе, а результат использовать в качестве программы для обработки числа n». Однако если мы попытаемся проделать это на практике, то в подавляющем большинстве случаев программа либо зависнет, либо выдаст непредсказуемый результат.

Дело в том, что в примере неявно используется соответствие между программными и данными $\xi : P \to \mathbb{N}$, где P — множество конечных последовательностей применения программы p самой к себе, $\mathbb{N}$ — натуральный ряд. Тогда правильная запись суперпозиции из примера будет иметь вид $n + 2 = [\xi^{-1}\, p\, \xi\, p](n)$, или в общем виде

$$n + x = [\xi^{-1} \underbrace{p \ldots p}_{x-1} \xi\, p](n). \tag{27.2}$$

Таким образом, даже завершающиеся программы не сводятся к полугруппе отображений, если разрешено использование одной программы в качестве входных данных для другой. Это не было замечено ранее, поскольку алгебры программ использовались в разработке электронных схем, где такое действие невозможно.

Замечание 1. *Здесь необходимо заметить, что в соответствии с теоремой Кона-Ребане [7] любая неассоциативная алгебра*

вложима в подходящую полугруппу. Поэтому может возникнуть вопрос — существует ли программный интерпретатор T, такой что $x * f = x \circ (f \circ T)$, то есть не является ли добавление неассоциативной операции искусственным, чего можно было бы избежать, если рассматривать программу вместе с интерпретатором. В общем случае на этот вопрос можно ответить отрицательно из следующих соображений: если программа принимает на вход данные ограниченной длины и размер кода этой программы превышает это ограничение, то взятие программы вместе с интерпретатором уменьшить её размер никак не сможет.

Алгебраический вычислитель

Абстрактные преобразователи сигналов, позволяющие реализовать алгебраический подход к вычислениям на практике, были предложены в работе Н. Н. Непейводы [8]. Алгебраический вычислитель, построенный на преобразователях такого типа, функционирует следующим образом:

1. Производится настройка конфигурации преобразователей, устанавливаются связи между ними;
2. Получается решение — путём распространения входного сигнала в сконфигурированной системе.

Поскольку состояние преобразователей не меняется в процессе вычислений, будем называть такие преобразователи статическими или «прямоточными». Одним из вариантов физической реализации вычислений на статических преобразователях являются кристаллические вычисления [9, 10].

Кроме статических также возможны динамические преобразователи, меняющие свою конфигурацию под воздействием входящих сигналов. Однако использование внутреннего состояния в процессе вычислений приводит к образованию «стены памяти», это то, чего хотелось бы избежать, поэтому далее мы не будем рассматривать такие преобразователи.

В свою очередь, алгебраический вычислитель на статических преобразователях ограничен тем, что не может производить действия над действиями в силу неизменности его конфигурации в процессе вычислений [8], другими словами, он не реализует вычисление операции «*» программных алгебр. Эта операция выполняется на подготовитель-

ной стадии (п. 1 списка), когда внешним устройством в соответствии с определённым алгоритмом производится настройка конфигурации алгебраического вычислителя.

Таким образом, любой алгоритм для алгебраического вычислителя распадается на две части:

1. Мета-алгоритм, выполняемый мета-вычислителем с целью конфигурации алгебраического вычислителя;

2. Собственно (прямоточный) алгоритм вычисления значения алгебраическим вычислителем.

В качестве ближайшей аналогии можно рассмотреть программу, реализующую вычисление некоторой функции, — алгоритм; и загрузчик, размещающий её в памяти и передающий управление, — мета-алгоритм.

Выводы

Направление алгебраических вычислений относится к тому новому, что является отчасти «хорошо забытым старым». Добавление в аксиоматику программных алгебр неассоциативной операции существенно расширило возможности подхода, в связи с этим открылись новые перспективы и возможности развития.

В частности, в данной работе установлено существование двух стадий работы алгебраического вычислителя: стадии конфигурации вычислителя, когда мета-вычислитель выполняет настройку алгебраического вычислителя в соответствии с мета-алгоритмом, и собственно стадии вычисления значения алгебраическим вычислителем путём распространения входного сигнала в физической системе заданной на первой стадии структуры.

Для дальнейшего исследования можно сформулировать задачу анализа вычислительной сложности мета-алгоритма в зависимости от возможностей конфигурирования, допускаемых алгебраическим вычислителем как физической сиситемой, оптимального соотношения между вычислительной сложностью мета-алгоритма и получаемого основного алгоритма, которое позволяло бы получить результат за минимальное полное время работы системы (в случае, когда существует несколько алгоритмов его вычисления).

Благодарности

Автор выражает признательность Н. Н. Непейводе за ценные советы и содействие в формулировке задачи, а также А. Н. Пустыгину, в беседе с которым родилась идея двух стадий работы алгебраического вычислителя.

Литература

[1] *Ландауэр Р.* Необратимость и выделение тепла в процессе вычислений. Перевод Чередникова И. О., Холмской А. Г., Квантовый компьютер и квантовые вычисления, 1999, **2**, с. 9–32.

[2] *Глушков В. М.* Синтез цифровых автоматов. — М.: Физматгиз, 1962. — 476 с.

[3] *Maurer W. D.* A theory of computer instructions. — Journal of the ACM, 1966, №13(2). — p. 226–235.

[4] *Глушков В. М.* Абстрактная теория автоматов. — УМН, 1961, **16**, №5(101), с. 3–62.

[5] *Непейвода Н. Н.* От численного моделирования к алгебраическому, Труды VI Международной конференции «Параллельные вычисления и задачи управления». — Москва, 24–26 октября 2012 г. (PACO-2012), Ин-т проблем управления им. В. А. Трапезникова РАН, **1**, с. 93–103.

[6] *Непейвода Н. Н.* Алгебраический подход к управлению, Проблемы управления, 2013, №6, с. 2–14.

[7] *Курош А.Г.* Общая алгебра. — М.: Физматлит, 1970. — 162 с.

[8] *Непейвода Н. Н.* Развитие алгебраического подхода к управлению: алгебры композиций и вставок. — XII Всероссийское совещание по процессам управления, Труды, М.: ИПУ РАН, 2014, с. 2587–2593.

[9] *Непейвода Н.Н., Цветков А.А., Хаткевич М.М.* Опыт проектирования кристаллографических вычислительных элементов. — Национальный Суперкомпьютерный Форум (НСКФ), Переславль-Залесский, ИПС РАН, 2015.

[10] *Демидов А.А.* Возможности вычислений на кристаллах. — Программные системы: теория и приложения: электронный научный журнал, 2015, **6**, №4 (27), с. 353–358.

Сергей Перепелов
Ставрополь, МБОУ СОШ № 34 города Ставрополя

Проект: QSquidClassRoom `http://qsquidclassroom.sourceforge.net`

QSquidClassRoom — простое управление прокси-сервером squid3

Аннотация

QSquidClassRoom — это простая программа с графическим пользовательским интерфейсом, предназначенная для управления прокси-сервером squid3 в компьютерном классе. Программа не является универсальной графической оболочкой для squid3, но ориентирована на максимально простое управление прокси-сервером без привилегий суперпользователя в рамках заданной конфигурации. Особое внимание уделено вопросам безопасности. Программа предназначена прежде всего для использования на учительских компьютерах в компьютерных классах, однако легко может быть настроена и для других задач.

QSquidClassRoom — это приложение, которое позволяет управлять конфигурацией прокси-сервера squid3 быстро и интуитивно. Предпосылкой к созданию этой программы и, по сути, постановкой задачи явилась следующая ситуация: имеется школьный компьютерный класс, в котором учительский компьютер работает под управлением ОС Linux. Ученические компьютеры могут работать под управлением других систем. Требуется предоставить ученическим компьютерам выход в Интернет под полным контролем учителя (подразумеваются возможности заблокировать доступ в сеть некоторым или всем ученикам, ограничить или предоставить доступ к тем или иным ресурсам сети, ограничить максимальную скорость для клиента, ограничить максимальное число подключений и т. д.). Для решения этой задачи подходит прокси-сервер squid3. Однако есть две проблемы. Во-первых, для управления прокси-сервером в большинстве систем требуются привилегии суперпользователя. Во-вторых, управление через командную строку или web-интерфейс не очень удобно в ходе урока, т. к. требует достаточно большого количества операций.

QSquidClassRoom призван решить обе этих проблемы. Во-первых, команда, принуждающая прокси-сервер перечитать конфигурационный файл и списки доступа, вынесена в отдельный исполняемый файл и средствами подмены идентификаторов обеспечивает возможность

непривилегированному пользователю влиять на поведение прокси-сервера, давая ему лишь необходимый минимум прав. Во-вторых, графический интерфейс программы построен таким образом, чтобы обеспечивать выполнение типовых операций одним или двумя щелчками мыши. Также предусмотрены возможности группировки ресурсов по меткам, применения фильтров к ресурсам, поддержка технологии drag&drop и многое другое. Графический интерфейс ориентирован на возможность управления программой не только при помощи мыши, но и, например, с интерактивной доски. Вид окна программы (главного и единственного) представлен на рисунке:

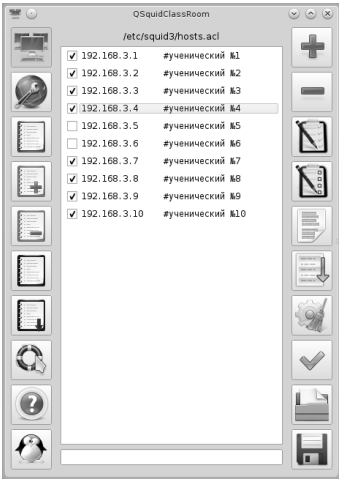


Рис. 1: Главное окно QSquidClassRoom

В смысле взаимодействия с прокси-сервером программа позволяет управлять списками доступа (до семи списков), но не затрагивает основной конфигурационный файл прокси-сервера. Возможно изменение смысловой нагрузки, возлагаемой на каждый из списков.

Особое внимание уделялось безопасности функционирования программы, которая строилась на трех основных принципах:

- надёжное функционирование кода;
- защита программы от подмены переменных окружения, переполнений буфера и некорректных входных данных;

- разграничение прав на настройку программы.

В коде программы обрабатываются практически все исключительные ситуации и возвращаемые значения вызываемых функций. Соответствующие сообщения при необходимости выдаются пользователю или направляются в стандартный поток ошибок.

Также проверяются все передаваемые в программу её параметры. Программа использует только полные пути при обращении к другим программам и защищена от подмены переменных окружения, т. к. не доверяет им в серьёзных вопросах. Также предусмотрена защита от чтения собственных конфигурационных файлов слишком большой длины (например если вместо реального конфигурационного файла передаётся бесконечный, например, случайно генерируемый поток данных). Аналогичные меры приняты и в отношении обрабатываемых списков доступа. Все принимаемые программой входные данные проверяются на соответствие предъявляемым к ним требования ещё на этапе чтения конфигурационного файла.

Настройка QSquidClassRoom разделена на три уровня: конфигурация по умолчанию, встроенная в код программы, конфигурация, заданная системным администратором, и пользовательская конфигурация. Настройки администратора переопределяют практически все настройки пользователя (за исключением нескольких безопасных с системной точки зрения). Настройки по умолчанию обеспечивают приемлемое поведение программы в условиях отсутствия или некорректности конфигурационных файлов. Отсутствие некоторого параметра в системной конфигурации, в то же время, позволяет передать управление этим параметром пользователю.

Множество настраиваемых параметров, а также тот факт, что взаимодействие с прокси-сервером вынесено в отдельный исполняемый файл открывает возможность для использования QSquidClassRoom не только для управления squid3.

В общем, при разработке кода программы учитывались все аспекты, перечисленные например в [1].

Программа распространяется вместе с подробной документацией на русском и английском языках в исходных кодах и в формате пакетов Debian для архитектур i586 и amd64. Корректность и удобство функционирования программы проверены 18 месяцами её эксплуатации в трёх компьютерных классах двух образовательных учреждений, где она работает под управлением ОС Debian 8 и Kubuntu

14.04. Рассматриваемая здесь версия является уже третьей, содержащей значительные улучшения с точки зрения безопасности, удобства интерфейса и потребления ресурсов по сравнению с предыдущими версиями.

В качестве дальнейшего направления развития рассматривается перерождения проекта в универсальную программу для управления списками, аналогичными спискам доступа squid3 со специализацией программы средствами конфигурационных файлов с сохранением простоты и удобства интерфейса для конечного пользователя.

Литература

[1] *David A. Wheeler*, Secure Programming for Linux and Unix HOWTO, 2004

Сергей Мартишин, Марина Храпченко
Москва, Институт системного программирования РАН

Большие данные: безопасность, надёжное хранение и распределенные вычисления

Аннотация

Рассмотрены основные технологии на основе СПО (NoSQL, MapReduce и Hadoop) для работы с большими данными и организации распределённых вычислений над ними.

Современное общество оперирует значительно большими объёмами информации, нежели десять, а тем более двадцать лет назад. Соответственно растут требования к надёжности её хранения и скорости её обработки, что в свою очередь привело к изменению используемых технологий. Особенно это касается так называемых «Big Data» («Большие данные»), то есть такого огромного объёма разнородной информации, обработка которого стандартными методами за разумное время практически невозможна. В этой связи представляется важным изучение студентами и магистрантами, обучающиеся по направлениям «Информационные системы и технологии», «Информатика и вычислительная техника», «Программная инженерия», технологий, предназначенных для работы с большими данными, основанных на свободном и свободно распространяемом программном обеспечении (СПО).

В настоящее время для организации хранения большого объёма данных применяется горизонтальное масштабирование (увеличение числа серверов), а в качестве основного инструмента обработки данных используются распределённые вычисления, которые позволяют производить операции с масштабируемыми данными на различных узлах одновременно.

Говоря о надёжности следует заметить, что для программного обеспечения (ПО) помимо требований сохранения уровня качества ПО, также предъявляются требования к устойчивости ПО к отказам и способности его к восстановлению [5].

С понятием надёжности тесно связано понятие безопасности, поскольку если хранимые данные подвергаются злоумышленному воздействию, это неизбежно приводит к снижению уровня качества ПО. Как правило в любых современных средствах хранения и обработки информации встроены средства защиты. Если необходимо обеспечить безопасность данных сверх стандартных средств защиты, то эта задача требует иных подходов и более подробно рассматривается авторами в [1].

В качестве основных технологий применяются NoSQL, MapReduce и Hadoop. Заметим, что все эти технологии являются СПО.

Основной целью разработки NoSQL-систем была возможность горизонтального масштабирования, которое для традиционных SQL-систем являлось достаточно трудной задачей. Традиционно СУБД NoSQL предлагают технологии репликации (replication — размещение и обслуживание серверов базы данных на нескольких физических узлах, целью которого является обеспечение избыточности и обработки отказа, а также увеличение доступности и сбалансировать данных, в основном при чтении) и шардинга (sharding — сегментирование базы данных и размещение сегментов на различных серверах. Это, например, особенно актуально в тех случаях, когда объём базы данных не позволяет организовать хранение данных из-за ограниченных возможностей адресации. Или, например, сервер не справляется с нагрузкой и снижается эффективность работы пользователей). Для более надёжной работы в условиях реальной эксплуатации масштабируемых баз данных первоначально выполняется репликация базы данных, а далее производится сегментирование [2,4].

Hadoop — фреймворк, основной задачей которого является распределённая обработка больших массивов данных, был разработан Apache Software Foundation, поэтому основным дистрибутивом явля-

ется Apache Hadoop. Именно эту открытую платформу предлагается использовать студентам на практике.

В основе Hadoop лежит Hadoop Distributed File System (HDFS — распределённая файловая система, предназначенная для хранения данных большого объёма), необходимые библиотеки и утилиты (Hadoop Common), а также ПО для управления заданиями и ресурсами кластера (Hadoop YARN) [3].

При использовании HDFS исходные файлы большого объёма разбиваются на файлы меньшего объёма (по умолчанию — 64 мегабайта), которые хранятся на нескольких недорогих машинах, причем для обеспечения надёжности HDFS реплицирует все поступающие данные (по умолчанию в тройном объёме) и хранит их на различных машинах.

Поскольку HDFS предназначена для хранения данных, но не для их обработки, то необходим некий инструмент, позволяющий это делать. Apache Hadoop имеет встроенный движок Hadoop MapReduce, написанный на Java и основанный на модели распределённых вычислений MapReduce, который позволяет эффективно работать с HDFS. Заметим, что для работы с HDFS могут применяться и иные движки, равно как и технология MapReduce может применяться при иной организации данных.

Технология MapReduce предназначен для работы с большими данными и состоит из двух шагов: Map (предварительная обработка данных) и Reduce (свёртка предварительно обработанных данных). Функция Map получает исходные данные (ключ и связанные с ним данные) и производит их предварительную обработку — генерирует пары (ключ, значение). На каждом узле кластера производится обработка своей части данных (заметим, что на одном и том же узле одновременно может выполняться несколько Map-процессов). Функция Reduce используя выходные данные функции Map производит свёртку и возвращает итоговый список значений. Большим преимуществом технологии является то, что она может надёжно работать на платформах с низкими показателями надёжности. Это достигается за счёт реализации алгоритма обнаружения потенциальных сбоев и обработки отказовых ситуаций путем распределения операции обработки данных по всем узлам сети, а также реализации алгоритма.

Hadoop MapReduce является фреймвоком, который упрощает создание приложений для распределённой обработки больших объёмов (терабайты) данных на больших кластерах (тысячи узлов). Заметим,

что технология группировки MapReduce может быть применена к различным хранилищам данных, в том числе, например, к MongoDB.

Студентам для освоения материала предлагается цикл лабораторных работ на СУБД NoSQL MongoDB, которая позволяет на локальном компьютере произвести репликацию и шардинг и проверить надёжность функционирования СУБД при отключении сервера, а также выполнить серию заданий на распределённые вычисления с помощью Apache Hadoop и Hadoop MapReduce и изучить встроенные средства безопасности.

Литература

[1] *Варновский Н. П., Мартишин С. А., Храпченко М. В., Шокуров А. В.* Пороговые системы гомоморфного шифрования и защита информации в облачных вычислениях. // «Программирование», N4, 2015, с. 47–51.

[2] *Фаулер М., Садаладж П. Дж.* NoSQL: новая методология разработки нереляционных баз данных — NoSQL Distilled. — М.: «Вильямс», 2013. — 192 с. — ISBN 978-5-8459-1829-1

[3] Hadoop. [Электронный ресурс]. Режим доступа: URL: `http://hadoop.apache.org` — Яз. англ. Дата обращения: 25.12.2015.

[4] MongoDB. [Электронный ресурс]. Режим доступа: URL: `http://www.mongodb.org` — Яз. англ. Дата обращения: 30.11.2014.

[5] ГОСТ Р ИСО/МЭК 9126-93 Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.

Дочкин Сергей
Кемерово, ФГБОУ ВПО «Кузбасский государственный технический университет»

Особенности применения свободных продуктов в вузовской практике

Аннотация

В статье рассматривается опыт использования программного обеспечения на основе свободных лицензий в системе высшего образования на примере технического вуза, связанные с этим трудности и результаты проведенных исследований. Данные вопросы рассматриваются в аспекте использования СПО в образовательных организациях высшего образования и дополнительного профессионального образования, решающих задачи подготовки кадров для экономики и органов управления региона. Отмечается роль образовательных организаций дополнительного профессионального образования, решающих задачи повышения квалификации и профессиональной переподготовки педагогов вузов и представителей органов власти задачи повышения квалификации и профессиональной переподготовки педагогов.

Развитие отечественной отрасли информационно-коммуникационных технологий (ИКТ) является одним из стратегических направлений развития современного российского государства и общества. В тоже время несмотря на высокие темпы развития ИКТ, которые были показаны в предыдущие годы, проблема отставания России по показателям, характеризующим состояние информационного общества и степень информатизации остается нерешенной. Более того, остаются не выполненными ряд концептуальных решений, срок исполнения которых закончен. К примеру, согласно Плана реализации Стратегии развития информационного общества (СРИО) доля отчественных товаров и услуг в обхъеме внутреннего рынка информационных и телекоммуникационных технологий к концу 2015 года должна достигать более 50%. Следует отметить, что в данных документах особая роль отводилась и свободному программному обеспечению (СПО). Согласно Плана реализации СРИО уже к концу 2010 года 25% общеобразовательных учреждений каждого региона РФ должны были использовать СПО не менее, чем на 50% компьютерах, однако на январь 2016 года проведенное нами исследование показало, что данный показатель все еще не достигнут. Разумеется внедрение

СПО не сможет решить всех проблем в информационной сфере страны, однако, на наш взгляд способно решить два важные задачи: добиться экономии средств и вывести на иной, более высокий, уровень ИКТ-компетентности наших специалистов и выпускников вузов. Не надо забывать, что в связи событием последних лет разработка отечественного программного обеспечения в целях обеспечения технологической независимости страны от зарубежных поставщиков программного обеспечения так же рассматривается как важная задача. Впрочем как и для решения вопроса экономии бюджетных средств, особенно посредством расширение использования в органах власти (государственной и муниципальной), государственных учреждениях и организациях РФ свободного программного обеспечения.

Правительством были предприняли меры по внедрению СПО в государственные и образовательные структуры, однако признать их успешными сложно. К таким мерам можно отнести принятие некоторых нормативных документов по преимущественному использованию СПО в органах государственной власти, выпуску программного обеспечения под свободной лицензией за счет бюджета, создание государственных фондов программного обеспечения для государственных структур, включающих СПО и программное обеспечение, разработанное по заказу разных государственных организаций, мероприятия по популяризации СПО, массовое обучение населения СПО и т.д. Однако данные мероприятия не приобрели системный характер. К примеру, принятое пять лет назад на федеральном уровне Постановление Правительства РФ № 2299 от 17.12.2010 года по переходу федеральных органов исполнительной власти и федеральных бюджетных учреждений на использование свободного программного обеспечения к 2015 году практически и осталось не реализованным и неизвестным за пределами столичного региона. По крайней мере в регионах нет информации о разработке и применении пакета базового СПО для решения типовых задач деятельности органов исполнительной власти и внедрение его в подведомственных бюджетных учреждениях, не отработан механизм ежеквартального обновления, отсутствует информация о создании единого репозитория СПО, используемого в федеральных органах власти. Возможно, в отдельных министерствах и ведомствах развернуты информационные системы, использующие принципы СПО, однако данная практика не нашла распространения и как правило касается «силовых структур» и ведомств, связанных

с безопасностью страны. Однако это противоречит самим принципу открытости и доступности СПО, хотя и считается допустимым.

Проведенное нами исследование практики использования служащими федеральных органов исполнительной власти показало, что 89,7% из них используют проприентарные программные продукты, закупка которых которые финансируются из бюджета. И только чуть более 10 % используют в своей работе программы под свободной лицензией, при этом как правило это или офисные продукты, или специфические программы, которые требуются для решения определенных задач. Так же мы обратили внимание на тот факт, что большинство пользователей СПО в этом случае выбрали данные продукты из своего прошлого опыта профессиональной деятельности, и попытки переобучения служащих даже не в организациях предпринимались. Таким образом, персонал организаций не знает и не имеет возможности изучать данные продукты.

Соответственно, на наш взгляд проблема стоит в том, что знакомство и использование программ СПО пользователями как правило в 95% случаев завершается в момент окончания обучения в школе, так как в последующем, при обучении в вузах страны, как правило, абитуренты и студенты используют проприентарное программное обеспечение (ППО). Исследование показало, что 83,6% абитуриентов вуза предпочитают использовать Windows-ориентированные приложения, и количество пользователей растет в ходе обучения — почти 92% выпускников вуза перед выпуском считаю более знакомым именно пакеты ППО. В тоже время почти 46,8% студентов знакомы с операционной системой Android, и не испытывают затруднений в ее использовании на своих мобильных устройствах. Данный факт говорит о том, что причина малой доли распространености СПО в вузе не сколько в самих пользователях или в недостатках данного ПО, а в отсутствии целостной информационно-обучающей среды, которая бы функционировала на Linux-системах или ином СПО. При этом следует подчеркнуть, что внедрение СПО в систему образование должны быть нераздельно с процессами внедрения СПО в органы государственной власти, и именно в этом аспекте нами и будет проводиться исследование. Отрыв одного от другого только приведет к несогласованности в действиях, нарушению координированной работы, увеличению затрат и дискредитации самого смысла использования СПО.

К решению проблемы, по нашему мнению, следует приступать с организаций образования, обучая как преподавателей, так и самих

обучающихся. В соответствии с этим в техническом университете были проведены работы по внедрению использования некоторых продуктов СПО в деятельность студентов, причем придерживаясь следующей последовательности: 1 этап — использование кроссплатформенных программ, работающих под Windows (Firefox, OpenOffice.org, LibreOffice, GIMP и др); 2 этап — использование операционных систем GNU/Linux частично на компьютерах в компьютерных классах; 3 этап — преимущественное использование программ СПО в учебной и внеаудиторной деятельности.

Однако большей проблемой оказалось не переориентирование студентов, будущих выпускников, а обучение профессорско-преподавательского состава вуза для использования програмных продуктов под свободными лицензиями. Причем если с офисными программными продуктами (OpenOffice.org, LibreOffice), браузерами (Mozilla Firefox), графическими редакторами (GIMP и др.) особых проблем не было, то переход на операционные системы под GNU/Linux натолкнулся на определенные трудности. Причем причина была не в сложности освоения или недостатка времению. Требовалось преодолеть психологический барьер и определенный консерватизм пользователей, особенно в условиях, когда у подавляющего большинства и на домашних компьютерах и на рабочих местах на кафедра была установлена лицензионная операционная система под Windows. Пока СПО используют всего не более 10% педагогов. Проведенное исследование показало, что только около 13 % кафедр институтов нашего университета пользуются операционными системами на основе GNU/Linux. Свободные приложения Firefox, OpenOffice.org, GIMP и прочие уже освоены и установлены не менее чем на 7-8% компьютеров (в том числе и под ОС Windows).

Для более системной работы по подготовке ППС к использованию программных продуктов СПО, были скорректированы дополнительные профессиональные программы повышения квалификации для преподваталей кафедр и работников отделов и служб университета. В рамках данных программ ППС предлагалось освоить наиболее распространные и доступные офисные программы и графические редакторы, причем одна из целей было желание донести по слушателей (преподавателей вуза) что продукты СПО по своей функциональности не уступают подобным из семейства Windows.

Одной из самых популярных тем в перечне продуктов СПО, которые были востребованными ППС, стала изучение системы электрон-

ного обучения Moodle, которая была выбрана как основная для развертывания в КузГТУ. Установленная на сервере и интегрированная с образовательным порталом вуза данная система показала себя достаточно эффективным средством обучения студентов очного и заочного обучения. Для ее освоение в университете на постоянной основе в течении уже двух лет обучаются преподватели кафедр. К примеры в 2015 году были проведены курсы повышения квалификации по дополнительной профессиональной программе «Электронное обучение в Moodle: создание курса и организация обучения». Цель этих курсов — совершенствование информационно-коммуникационных компетенций у ППС вуза, необходимых для создания образовательных ресурсов, организации учебного материала на основе эффективного использования дистанционных образовательных технологий и системы электронного обучения университета на основе системы Moodle. Программа предполагала освоение 6 учебных модулей, из которых один модуль — теоретический, четыре — практических, еще один — итоговый. В ходе обучения слушатели познакомились с системой электронного обучения Moodle, развернутой в университете, сформировали умения по использованию программных средств оболочки, выработали первичные навыки в работе с элементами и ресурсами для создания обучающего курса. В ходе практических заданий каждый из слушателей создал свой первый курс в данной системе, в рамках итогового проекта — краткий курс по собственной дисциплине.

Такой целенаправленный подход позволил в краткий срок большому количеству преподавателей (около 45%) освоить данную систему и приступить к проектированию собственных курсов в рамках основных образовательных программ высшего образования.

Сделать предстоит еще очень много, однако мы понимаем, что использование СПО в вузовской практике это не самоцель, а попытка подготовить выпускника к продуктивной профессиональной деятельности в будущем, в информационно-насыщенной высокотехнологичной среде, чтобы выпускник мог легко ориентироваться в тех или иных информационных системах и программных продуктах. В сложившейся ситуации в наших планах — дальнейшее повышение квалификации работников вуза и ППС в области ИКТ (делая акцент на СПО), разработать механизм информирования о существующих разработках, а также разработать ряд модульных методик, программ и курсов, направленных на более активное использование свободно распространяемого ПО в вузовской практике.