АНО «Институт логики, когнитологии и развития личности»
ALT Linux

Седьмая конференция разработчиков свободных программ

Переславль, 26–27 июля 2010 года

Тезисы докладов

Москва,
Институт Логики,
2010

В книге собраны тезисы докладов, одобренных Программным комитетом Седьмой конференции разработчиков свободных программ. Круг рассматриваемых тем весьма широк: от новейших системных и прикладных разработок до правовых и экономических проблем, вопросов организации работы в проектах и аналитики.

Программа конференции

26 июля

11.30–12.30 Регистрация, кофе, чай

Дневное заседание
12.30–14.30

14.30–15.30 Обед

Вечернее заседание
15.30–19.00

27 июля

Дневное заседание
10.00–14.00

Вечернее заседание
15.00–19.00

Вне программы

Владимир Сухомлин
Москва, МГУ

Итоги реформы высшей школы

Аннотация

В докладе подводится итог реформы системы высшего профессионального образования в России, в частности, анализируется соответствующее законодательное обеспечение, результаты перехода на уровневую систему образования и образовательные стандарты третьего поколения, а также другие инициативы министерства образования и науки РФ, в совокупности производящих деструктивный эффект по отношению к национальной высшей школе, совсем недавно еще считавшейся одной из лучших в мире. В тезисах рассматриваются только ключевые моменты этой проблемы.

Введение

Россия, как и другие развитые страны, вступила в период перехода от постиндустриального общества к обществу, основанному на знаниях. Знание становится одним из ведущих сегментов производства, накопления и товарного обмена, ведущим «инструментом» в экономической деятельности человека. Процесс труда в таком обществе базируется, в основном, на потреблении знаний и сопровождается созданием новых знаний.

Система высшего образования в этих условиях превращается в одну из ведущих самостоятельных производящих отраслей экономики (не сферы обслуживания), обеспечивающую приращение человеческого капитала и, тем самым, приращение ВВП в инновационной экономике.

Все больше и больше стран (США, Китай, некоторые европейские и азиатские страны) ставят в качестве национальной стратегической задачи всеобщее высшее образование и создание системы образования для взрослых на протяжении жизни. Очевидно, что эта задача должна стать одной из важнейших и для России, если она претендует на достойное место в мировой экономике.

Во всемирном докладе ЮНЕСКО «К обществам знания» 2005 г. понятия «обучающиеся общества» и «образование для всех на протяжении всей жизни» возведены в ранг ключевых принципов построения новой модели социума — «обществ знаний». Каково же с этой точки зрения состояние российской системы высшего профессионального образования (ВПО), уже можно сказать, после состоявшейся реформы?

Высшая школа как мишень реформистской агрессии

Напомню, что путеводной звездой для реформаторов системы образования в России стали фетишизированные до абсолюта идеи болонизации высшего образования (по этому поводу автор высказывал свою позицию, нашедшую широкое признание среди коллег). Для достижения «этих высот» были определены следующие основные цели реформирования национальной системы высшего профессионального образования (ВПО):

1) переход на уровневую систему ВПО (от ступенчатой!);
2) и переход на федеральные государственные образовательные стандарты (ФГОС) нового поколения (третьего) от государственных образовательных стандартов (ГОС) второго поколения.

Реализация перехода на уровневую систему ВПО и ФГОСы третьего поколения началась с подведения под реформы соответствующей законодательной базы, основу которой ставили следующие Федеральные законы:

1. Федеральный закон от 24.10.2007 г. № 232-ФЗ «О внесении изменений в отдельные законодательные акты Российской Федерации (в части уровней высшего профессионального образования).
2. Федеральный закон от 01.12.2007 г. № 309-ФЗ «О внесении изменений в отдельные законодательные акты Российской Федерации в части изменения понятия структуры и содержания государственного образовательного стандарта».
3. Федеральный закон от 01.12.2007 г. № 307-ФЗ «О внесении изменений в отдельные законодательные акты Российской Федерации в целях предоставления объединениям работодателей права участвовать в разработке и реализации государственной политики в области профессионального образования».

4. Федеральный закон от 10 ноября 2009 г. № 260-ФЗ «О внесении изменений в отдельные законодательные акты РФ в связи с принятием ФЗ „О МГУ им. М. В. Ломоносова и СПбГУ“».

Параллельно был запущен процесс создания ФГОС третьего поколения, который проводился (с 2006 по 2010 гг.) под жестким методическим руководством со стороны министерства образования и науки РФ (МОН), навязавшего высшей школе собственную концепцию образовательного стандарта.

На текущий момент запущены механизмы перехода на уровневую систему ВПО и новые федеральные государственные образовательные стандарты, обусловленные содержанием следующих статей закона:

- Статьи 4 Федерального закона от 24.10.2007 г. № 232-ФЗ;
- Статьи 6 Федерального закона от 10.11.2009 г. № 260-ФЗ.

Они предусматривают порядок перехода вузов по решению их ученых советов на уровневую систему ВПО и стандарты нового поколения, конечные сроки приема в вузы по ступенчатой системе ВПО (до 30 декабря 2010 г.). Также в этих статьях устанавливается эквивалентность квалификаций «бакалавр», «специалист» и «магистр», ступенчатой и уровневой систем ВПО.

Как правило, возникает вопрос: «Что же дает тогда переход от ступеней к уровням, не принимая во внимание большую работу юристов по переписыванию законов, ведь при переходе на уровневую систему ВПО сохранены эквивалентность квалификационных позиций (степеней) подготовки, т.е. формально переход не привносит ничего нового? Что стоит за этим переходом?».

На самом деле за этим стоит значительное сокращение числа специальностей с наукоемкой подготовкой (с 535 до 107, т. е. в пять раз!). Это есть не что иное, как силовое выдавливание из образовательного поля специальностей, во многих случаях ущербное, примеров тому множество. Все это ведет к массовому сокращению уровней профессиональной подготовки — замене 5 летнего обучения на 4-х летнее (надо сказать, что в мире складываются обратные тенденции — доказано, что увеличение сроков обучения на один год ведет к существенному увеличению ВВП). Все это также ведет к значительному сокращению объема учебной нагрузки для преподавателей и ухудшению их материального положения. Особенно значительный ущерб наносится

уникальным преподавательским технологиям (как это показано в [1] на примере факультета ВМК).

Теперь вернемся к ФГОС третьего поколения. Чем он отличается от ГОС второго поколения и что он дает системе ВПО?

В новой концепции МОН ФГОС представляет собой совокупность следующих требований (фиксированных аж на 10 лет!), обязательных при реализации основных образовательных программ (ООП): к структуре ООП, условиям реализации ООП (кадровым, финансовым, материально-техническим, и пр.), результатам освоения ООП (т. е. к ожидаемой компетенции выпускников, выраженной определением набора компетенций, своего рода профессиональных характеристик, закрепленных в стандарте на 10 лет). Требования к объему знаний в стандарте отсутствует. Получается, что в век построения общества знаний, когда весь мир только и занимается проектированием таковых, знания из системы российского образования, важнейшей социокультурной технологии передачи и развития знаний, изгнаны! Правда, описание содержания обучения при этом как бы должно быть отражено в описании ООП, но последняя не имеет никакой юридической силы, а ее концепция, соответствующая современным веяниям, так и не разработана. Следует еще отметить, что МОН всячески старалось навязать вузам при формировании компетенций ФГОС брать за основу компетенции профессиональных стандартов, т. е. требований, предъявляемых к работникам конкретной области трудовой деятельности, загоняя этим образования и науки под требования отсталой российской экономики (более подробно смотри [2]).

Что же получилось в итоге с ФГОС? ФГОС функционально уступает ГОС, т. к. не содержит важнейшей компоненты нормирования профессиональной подготовки — требований к содержанию обучения (В ГОС второго поколения содержание обучения нормироваловь на уровне 70% от общей учебной нагрузки). Компетенции как цели обучения без поддерживающего объема знаний представляются просто общими лозунгами, мало полезными для проектирования ООП. ФГОС обнуляют все методическое обеспечение ВПО (аккредитование, лицензирование, грифование, мобильность). Созданные стандарты трудно гармонизировать с международными стандартами, например, CC2005/2008. Они являются не технологичными, так как не охватывают многопрофильность обучения по направлениям, не поддерживают множественность стратегий обучения, не имеют технологии поддержки и сопровождения.

Более подробный анализ этих и других решений МОН РФ можно найти в статьях автора «Пустое множество. Ч.1, Ч.2», «Советская Россия» от 10.03.2008 г. или в статье «По улице, не ведущей к храму. Профессор В.А. Сухомлин о реформах высшего образования». Совет Ректоров. №3, 2008, с. 65-72. В. А. Сухомлин. Профессиональные стандарты и образование. Перпендикулярный взгляд, ВМиК МГУ им. Ломоносова, МАКС пресс. 2008, 80 с. (доступной по адресу `http://forums.vif2.ru/showthread.php?t=456`). «Высшее образование будет без содержания». 12/11/2008 Сетевое издание «Сегодня.ру» (`http://segodnia.ru/index.php?pgid=2&partid=45&newsid=6940`).

Еще об одной из последних инициатив МОН РФ — подготовке за спиной профессиональной общественности «антикризисного» соглашения с Майкрософтом. По существу это есть не что иное, как сдача под патронаж известной компании всей системы российского образования. Оно в частности является определенным противодействием более широкому внедрению в образовательную практику открытого программного обеспечения. Более подробно об этом написано в статье автора «Антикризисная экспансия. Грядет ли эра „Майкрософт“ в российском образовании?», Советская Россия от 12/02/2009.

Бессистемность и противоречивость проводимых реформ наносит непоправимый ущерб системе ВПО России. По оценкам экспертов отставания России в этой сфере оценивается в 15 и более лет.

Все больше людей убеждаются в деструктивности проводимых в образовании реформ, их ущербности для страны. Примером тому, что это мнение, доминирующее в сфере образования, разделяется теперь большинством работодателей, служат следующие выдержки из документа, разработанного Союзом ИТ-директоров (`http://forums.vif2.ru/showpost.php?p=2300&postcount=1`). Этот документ называется «Рекомендации в сфере образования и науки, использования информационных ресурсов, современных технологий связи и передовых информационных технологий (Выработаны на заседании экспертов 25 марта 2009 г. при участии Школы ИТ-менеджмента АНХ при Правительстве РФ и Института Современного Развития)».

Вот две выдержки из этого документа: «...Эксперты разделяют точку зрения о том, что одной из основных причин, тормозящих переход России на рельсы постиндустриальной экономики, является несвоевременная и полностью лишенная долгосрочных ориентиров и идей реформа образования, в том числе, образования в сфере использования ИКТ, оторванность российских научно-исследовательских

центров и вузов от проблематики бизнеса, а бизнеса от их проблематики...», «...Органам исполнительной власти... 7. Отказаться от тотального введения якобы «болонской» системы двухуровневого образования, низводящей большинство вузов на уровень техникумов и воспринимаемой в России, как стремление Москвы вернуть себе монополию на элитное образование...».

Выводы

Что же делать? Считаю необходимым добиваться следующего:

1. Проведения широкого обсуждения в обществе и профессиональной сфере итогов реформирования системы ВПО. Подготовки и проведения Всероссийского съезда работников образования и науки по этому вопросу.
2. Признания властью полного провала реформы системы образования и приостановки перевода ВПО на уровневую модель и стандарты нового поколения.
3. Привлечения к судебной ответственности реформаторов, нанесших непоправимый ущерб системе ВПО и стране.

Литература

[1] Сухомлин, В. А. Полная победа инноваций над российским образованием : (размышления российского профессора о реформах высш. образования) / В. А. Сухомлин // Вестн. Московского ун-та. Сер. 20, Педагогическое образование. — 2009. — № 1. — С. 16—40.

[2] Сухомлин, В. А. Профессиональные стандарты и образование. Перпендикулярный взгляд. М.: ВМиК МГУ им. Ломоносова, «МАКС-пресс», 2008, 80 с.

Дмитрий Варенов, Александр Кузнецов
Москва, ФГУ ГНИИ ИТТ «Информика»

Проект: Федеральный центр информационно-образовательных ресурсов
http://fcior.edu.ru

Использование свободного программного обеспечения в составе аппаратно-программного комплекса Федерального центра информационно-образовательных ресурсов

В настоящее время в области информатизации образования основное внимание фокусируется на проблемах создания эффективных электронных образовательных ресурсов (ЭОР). В соответствии с мировым опытом на смену текстографическим электронным продуктам приходят высоко интерактивные, мультимедийно насыщенные ЭОР. В последнее время также получили распространение открытые образовательные модульные мультимедиа системы (ОМС), объединяющие электронные образовательные ресурсы трех типов: информационные, практические и контрольные.

Тем не менее, учитывая географические условия нашей страны, телекоммуникационный доступ к образовательным ресурсам сильно затруднен. Таким образом для повышения доступности ЭОР для конечных пользователей (учителей и учащихся) был запущен проект федерального центра информационно-образовательных ресурсов (ФЦИОР), направленный на распространение электронных образовательных ресурсов и сервисов для всех уровней и ступеней образования. Федеральный центр информационно-образовательных ресурсов (ФЦИОР) представляет собой комплекс оборудования, электронных образовательных ресурсов (ЭОР), информационных систем и инструментальных средств, который обеспечивает практическую реализацию сервис-ориентированной модели информатизации сферы образования, интеграцию и унификацию разрозненных информационных, управленческих систем и электронных образовательных ресурсов. Портал ФЦИОР обеспечивает каталогизацию электронных образовательных ресурсов различного типа за счет использования единой информационной модели метаданных, основанной на стандарте LOM. Электронные учебные модули создаются по тематическим элементам

учебных предметов и дисциплин. Каждый учебный модуль автономен и представляет собой законченный интерактивный мультимедиапродукт, нацеленный на решение определенной учебной задачи. Для воспроизведения учебного модуля на компьютере требуется предварительно установить специальный программный продукт — ОМС-плеер.

В обеспечение беспрепятственной доставки ЭОР конечным пользователям в регионах, в настоящее время ФГУ ГНИИ ИТТ «Информика» (эксплуатирующая организация) инициировала пилотный проект по созданию региональных представительств ФЦИОР в пяти регионах — в Ставропольском крае (СевКавГТУ), Краснодарском крае (Куб. ГУ), Саратовской (СГТУ) и Томской (ТГУ) области, республике Башкортостан (БашГПУ). Данные организации предоставляют доступ для школьных образовательных учреждений к содержимому хранилища ФЦИОР посредством локализации регионального трафика. Целью данного проекта является обеспечение доступа для школьных образовательных учреждений к содержимому хранилища ФЦИОР посредством локализации регионального трафика. Такой подход позволяет сформировать федеральную межрегиональную ассоциацию профессиональных сетевых сообществ преподавателей и разработчиков ЭОР, поддерживающую обмен опытом в преподавательской практике использования ЭОР, коллективную разработку и апробацию методических материалов использования ЭОР в учебной деятельности. В ходе работ по созданию региональных представительств ФЦИОР были предоставлены серверные мощности, размещенные на базе государственных университетов, которые предоставляют региональным учреждениям общего образования безвозмездные услуги интернет-доступа к электронным образовательным ресурсам, размещенным в региональных представительствах ФЦИОР.

Порталы региональных представительств Федерального центра информационно-образовательных ресурсов, реализующих федеральную межрегиональную ассоциацию профессиональных сетевых сообществ преподавателей и разработчиков электронных образовательных ресурсов, в качестве основы используют следующее свободное программное обеспечение (СПО), функционирующее на базе свободной операционной системы:

1. Аппаратный комплекс представительств федерального центра информационно-образовательных ресурсов работает под управлением свободной операционной системы GNU/Linux.

2. Apache Tomcat — Используется в качестве сервера приложений и Web-сервера.
3. Apache Lucene — Библиотека полнотекстового поиска по карточкам-метаописаниям электронных образовательных ресурсов, реализует поисковый сервис порталов региональных представительств ФЦИОР.
4. Apache HTTP Server — Используется в качестве балансировщика нагрузки на фронтальных серверах региональных представительств ФЦИОР.
5. MySQL Server Community Edition — Реализует базу данных карточек-метаописаний электронных образовательных ресурсов, размещенных на порталах региональных представительств ФЦИОР.
6. VsFTPd — FTP-сервер, реализующий хранилище электронных образовательных ресурсов, размещенных на порталах региональных представительств ФЦИОР.

Было выполнено нагрузочное тестирование в течение четырёх месяцев непрерывной работы, на основе обеспечения одновременной работы пользователей с разделами каталога и поиска. Для каждого раздела определялось четыре варианта работы пользователей. Каждый вариант обеспечивал работу с разделом последовательностью нескольких запросов (первоначальная страница раздела, задание запроса, получение списка ресурсов, просмотр 3-х первых страниц результирующего списка, просмотр карточки, загрузка метаданных). Выбор раздела и варианта осуществлялся случайным образом. Имитировалась работа 1000 конкурентных пользователей. В результате, среднее время отклика системы, развернутой в кластере, состоящем из двух узлов, не превысило 1 секунды. Этот и другие полученные результаты нагрузочного тестирования дают основание утверждать о правильности выбранных технологических и программных решений.

Андрей Михеев
Москва, Консалтинговая группа РУНА
Проект: RunaWFE http://wf.runa.ru/rus

Проект RunaWFE — свободная система управления бизнес-процессами и административными регламентами. Новые возможности последних версий

Аннотация

В докладе рассказывается про новые возможности, появившиеся в 2010 г. в версиях 3.1 и 3.2: поддержка элемента Мульти-Действие и возможность архивирования экземпляров процессов

Описание системы RunaWFE

RunaWFE — открытая, масштабируемая, ориентированной на конечного пользователя система управления бизнес-процессами и административными регламентами. Система платформонезависима (написана на Java), распространяется под LGPL-лицензией, разрабатывается Консалтинговой группой РУНА.

Основная задача системы: раздавать задания исполнителям. Последовательность заданий определяется графом бизнес-процесса, который менеджер или бизнес-аналитик может быстро изменять при помощи редактора бизнес-процессов.

Система состоит из следующих компонентов:

- RunaWFE-сервер;
- Графический редактор процессов;
- Клиент-оповещатель о поступивших заданиях.

Функции компонента «RunaWFE-сервер»:

- Работа с определениями и экземплярами процессов;
- Работа со списками заданий;
- Визуализация форм, соответствующих заданиям;
- Работа с системой через web-браузер;
- Предоставление возможности работы с системой приложениям; специального вида - ботам. (В частности, боты могут моделировать работу сотрудника предприятия);

• Авторизация и аутентификация пользователей.

Функции компонента «Графический редактор процессов»:

• Редактирование графа процесса;
• Создание и редактирование графических форм заданий;
• Создание и назначение ролей;
• Создание переменных.

Функции компонента «Клиент-оповещатель о поступивших заданиях»:

• Оповещение о поступивших заданиях;
• Работа с системой через специальное приложение-клиент.

Возможные последовательности заданий бизнес-процесса определяет направленный граф: множество узлов, соединенных между собой линиями со стрелками — возможными переходами. Узлы бизнес-процесса могут быть двух типов — узлы, соответствующие шагам процесса (узлы-действия), и маршрутные узлы, иногда в литературе эти узлы называются — «вентили». По переходам перемещается точка управления (указатель на активный узел процесса), руководствуясь правилами в маршрутных узлах.

В узле-действии система дает задание исполнителю (сотруднику или информационной системе) и ждет ответа (сообщения, что работа выполнена). После ответа исполнителя точка управления движется по переходу к следующему узлу процесса.

Маршрутный узел соответствует разветвлению или разделению-слиянию точек управления. В таких узлах система выбирает на основании содержащихся в маршрутных узлах правил следующий узел (узлы), в который будет передано управление.

В выполняющемся бизнес-процессе одновременно может быть несколько точек управления. В соответствии с бизнес-логикой процесса точка управления в маршрутном узле может разделиться на несколько точек управления, также точки управления могут ждать друг друга в другом маршрутном узле и далее слиться в одну точку управления.

Элемент Мульти-действие

Мульти-действие — специальный вид узла-действия. В момент прихода управления в мульти-действие создается известное к этому моменту времени количество экземпляров этого узла-действия, для каждого узла организуется своя точка управления и далее для

Рис. 1. Обозначение конструкции «Мульти-действие»

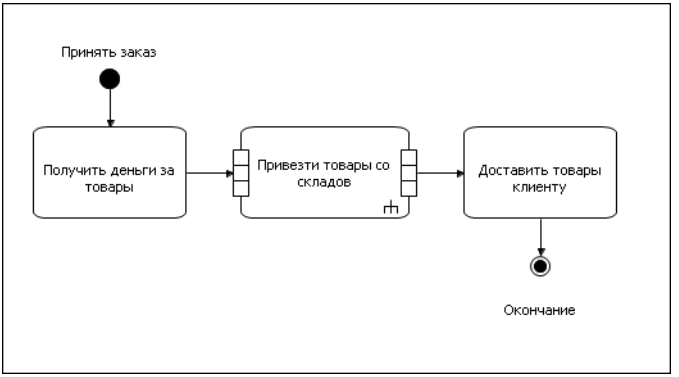

Рис. 2. Условный пример процесса с мульти-действием

каждой точки управления параллельно и независимо выполняются действия разных экземпляров этого узла. После того, как действия всех точек управления выполнены, управление переходит к следующему узлу по исходящему переходу.

Замечание. Количество экземпляров мульти-действия определяется значением специальной «присоединенной» переменной в момент прихода управления в этот узел-действие.

Поясним элемент «мульти-действие» на примере процесса оформления покупки набора товаров, хранящихся на нескольких складах. В этом процессе сотрудник принимает заказ клиента, потом система ждет прихода денег за товары, потом по списку заказанных товаров определяются склады, на которых эти товары находятся и для каждого склада, на котором находятся заказанные товары, параллельно запускаются процедуры получения этих товаров со складов. Причём, до заказа клиента неизвестно, какие склады будут задействованы в

мульти-действии. Неизвестно даже, сколько складов будет задействовано — все определяется списком товаров. После того как товары привезены со складов, они доставляются клиенту.

Архивирование экземпляров процессов

В версии 3.2 системы RunaWFE была добавлена возможность переноса старых, уже не используемых экземпляров бизнес-процессов в архив. Эта функциональность реализуется двумя командами: archivingOldProcessInstances и removeOldProcessInstances. Первая команда копирует экземпляр процесса в архив, вторая команда удаляет экземпляр процесса из рабочей системы.

Литература и ссылки

- OnLine demo системы RunaWFE доступно по адресу: `http://wf.runa.ru/Online_Demo`
- Ссылка на проект RunaWFE на портале sourceforge: `http://sourceforge.net/projects/runawfe`
- Ссылка на русскоязычный сайт проекта: `http://wf.runa.ru/rus`

Денис Медведев
Москва

Опыт построения информационной системы на основе Runa WFE

В ходе обычной сисадминской деятельности была получена задача опроса различных категорий людей. Оформление опросов планировалось другим отделом и создание форм тоже было поручено другим сотрудникам.

Были рассмотрены следующие пути решения этой задачи:

- Использование коммерческих и некоммерческих внешних решений (google docs, surveymonkey и др.). Этот вариант был отброшен по соображениям безопасности данных и требуемой гибкости решения.

- Использование собственных скриптов (php+mysql). Это решение требовало большого объема программирования и «изобретения велосипедов».
- Использование мощной системы поддержки обработки документов. Этот вариант и был выбран — в качестве системы поддержки документов-форм опроса была выбрана Runa WFE — свободный продукт от Runa consulting group.

Структура системы: Система логически устроена весьма просто и состоит для каждого опроса из единственного процесса системы Runa, включающего в себя одну или несколько форм. Эти формы заполняются пользователями системы. После заполнения последних форм процесс передается боту, который осуществляет заполнение полей базы данных полями опроса. Далее данные из этой базы экспортируются и используются заказчиками опроса путем экспорта в Excel через ODBC.

Система работает под ОС Fedora Core 12. В качестве базы данных используется MySQL. Формы подготавливаются заказчиком опросов. Связь с базой данных, создание переменных и программирование бота производится силами IT-отдела. Так как задача относительно редкая, то делать это приходится нечасто.

Система находится в рабочей эксплуатации, задача заказчика была выполнена за срок около двух недель, считая от постановки задачи.

Георгий Мартынов, Евгения Кушнир
Архангельск, ГОУ ВПО «Поморский государственный университет им. М. В. Ломоносова»

Разработка информационной системы анализа исполнения государственных контрактов отдела информатизации управления федеральной службы судебных приставов

Аннотация

Отдел информатизации постоянно заключает контракты и договора на поставку компьютерной техники и комплектующих, а также производит их утилизацию. В связи с этим наблюдается значительный

рост документооборота, автоматизация которого позволит снизить нагрузку по обработке соответствующих данных и подведению итогов состояния исполнения договоров.

Конечный продукт должен соответствовать следующим базовым требованиям:

- ИС должна систематизировать данные по техническим заданиям, государственным контрактам, договорам, счет-фактурам;
- ИС должна проводить анализ по различным параметрам (интенсивность ремонта и использования оргтехники, частоту заправки картриджей и т. д.);
- Данные, получаемые в результате обработки, должны быть представлены в удобном виде для последующего формирования запросов и анализа;
- Интерфейс должен быть адаптирован для удобного ввода, редактирования и просмотра информации в электронном виде.

В настоящее время деятельность любой организаций напрямую зависит от информированности и способности эффективно использовать имеющуюся информацию. Для этого необходимо провести большую работу по сбору, упорядочиванию, переработке информации, а также ее анализу. При большом количестве информации такая работа становится очень трудоемкой и требует большого количества времени и кадровых ресурсов. Вполне естественно, что внедрение компьютерных технологий должно позволить существенно увеличить производительность труда и снизить затраты. Поэтому, перед организациями на первом плане стоит задача внедрения современных информационных технологий.

Одной из таких организаций является Федеральная служба судебных приставов. Необходимостью создания автоматизированной системы для отдела информатизации данной службы, послужил постоянный рост объемов документации, а так же большие временные затраты на ручную обработку данных, анализ результатов исполнения государственных контрактов и договоров.

Следует отметить, что на данный момент времени в Службе судебных приставов не существует программных средств, автоматизирующих работу отдела информатизации по вышеизложенному направлению.

Исходя из этого, к конечному продукту предъявляются следующие требования:

- программа должна систематизировать данные по техническим заданиям государственным контрактам, договорам, счет-фактурам;

- программа должна проводить анализ по различным параметрам (интенсивность ремонта и использования оргтехники, частоту заправки картриджей и т. д.);
- данные получаемые в результате обработки должны быть представлены в виде удобном для формирования различных запросов.

Анализ изложенных требований приводит к выбору структуры продукта.

Так как задача будущего продукта — систематизировать данные и документы, то это означает, что работы проводимые в нем будут связанны не только с вводом, но и с накоплением информации. Поэтому логичнее всего использовать в качестве накопителя и хранителя данных базу данных. Кроме того, формирование БД лучше проводить под управлением какой либо СУБД, так как она берет на себя ответственность за поддержание ограничений целостности данных.

С учетом того, что силами отдела информатизации в Службе судебных приставов был развернут сервер под управлением СУБД Firebird Server 2.1, вполне разумно использовать СУБД Firebird для формирования БД.

После проведенного анализа данных и информационных потоков организации, была построена диаграмма сущность — связь, на основе которой сформированы таблицы и схема БД.

База данных BASE.FBD состоит из 9 таблиц, связанных между собой.

Таблица PODRAZDELENIYA используется для хранения данных о названиях подразделений ФССП по Архангельской области.

Таблица NAIM_IZD используется для хранения данных о наименовании изделий.

Таблица ISPOLN используется для хранения данных об исполнителях.

Таблица PREDM используется для хранения данных о предметах государственных контрактов.

Таблица VID_RAB используется для хранения данных о видах работ.

Таблица TEX_ZAD используется для хранения данных о технических заданиях для будущих государственных контрактов и договоров.

Таблица GOS_CONTR используется для хранения данных о государственных контрактах.

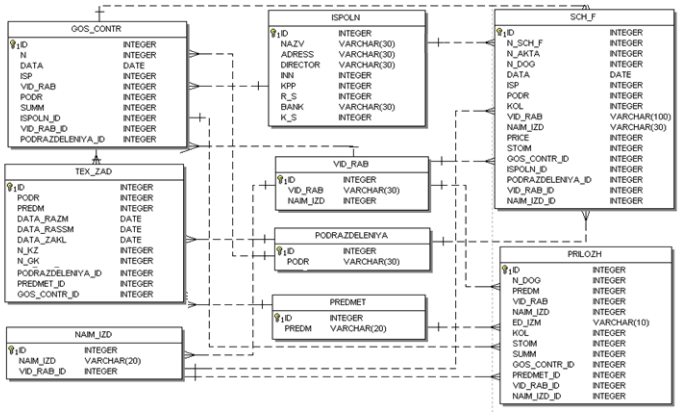

Рис. 1. Схема данных БД

Таблица PRILOZH используется для хранения данных о приложениях к государственным контрактам.

Таблица SCH_F используется для хранения данных о счет-фактурах.

Связи между вышеперечисленными таблицами представлены на рис. 1.

Для формирования клиентской части программы необходимо использовать дополнительную среду разработки приложений. Так как Firebird является свободным продуктом и, кроме того, является кроссплатформенным, то для поддержки данных свойств выбор пал на среду Lazarus 0.9.28.2. В настоящее время Lazarus практически полностью поддерживает виджеты Win32, GTK1, GTK2, Carbon. Среда разработки Lazarus интуитивно понятна, что в свою очередь обеспечивает высокую производительность труда разработчика. Язык разработки, используемый в среде Lazarus, — Free Pascal.

Подключение к базе данных будет обеспечиваться через компонент IBConnection, который также используется другими компонентами для подключения к источникам данных. Вызов необходимых форм будем осуществлять через пункты меню главной формы, появляющегося после подключения к базе данных.

Для удобства работы с компонентом TDBGrid реализуется метод позволяющий производить сортировку по клику на шапке таблицы. Кроме того, необходимо реализовать запрос «Исполнение государ-

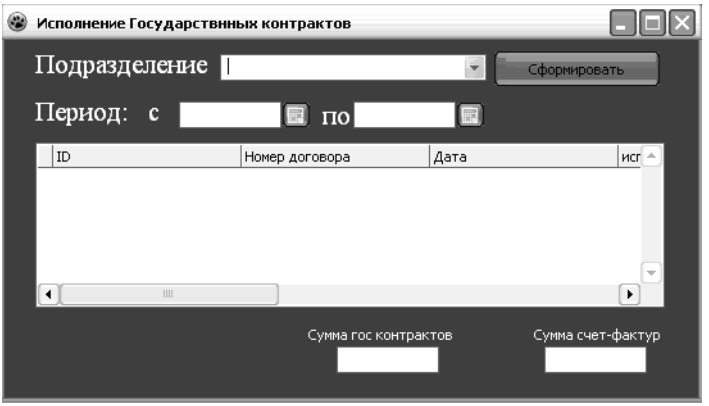

Рис. 2. Форма «Исполнение Государственных контрактов»

ственных контрактов», обеспечивающего отражение состояния исполнения контрактов и договоров.

В результате разработана система, позволяющая производить не только стандартные операции по вводу, накоплению и обработке данных, но и, за счет реализованного механизма, проводить аналитику исполнимости государственных контрактов с выборкой по подразделениям и периоду заключения контракта.

Так как печать отчета по результатам анализа исполнения государственных контрактов не ставилась в качестве задачи, то отражение статуса контракта формируется за счет порядка вывода в таблице DBGrid.

На сегодняшний день система проходит процесс тестирования и внедрения в отделе информатизации Управления Федеральной службы судебных приставов по Архангельской области.

Литература

[1] Положение об отделе информатизации и обеспечения информационной безопасности Управления Федеральной службы судебных приставов по Архангельской области.

[2] Рудюк С.А., Lazarus. Delphi-кросс-платформенный. Программирование для свободных людей. 2006. [Электронный ресурс]. — Режим доступа: `http://corp2.net`.

[3] Сайт Firebird [Электронный ресурс]. — Режим доступа: http://www.firebirdsql.org/.

[4] Официальный сайт Федеральной службы судебных приставов России. [Электронный ресурс]. — Режим доступа: http://www.fssprus.ru/.

[5] ИТ блокнот /Николай Войнов/ винегрет восприятия информационных технологий. [Электронный ресурс]. — Режим доступа: http://nvoynov.blogspot.com/2007/09/blog-post_19.html.

[6] TProPortal.RU. Использование SQLdb в Lazarus: основы работы с базами данных [Электронный ресурс]. — Режим доступа: http://www.itproportal.ru/sqldbinlazarus.html.

[7] FreePascal.ru. Информационный портал для разработчиков. [Электронный ресурс]. — Режим доступа: http://www.freepascal.ru/article/lazarus/.

Василий Колесниченко
Архангельск, ПГУ им. М. В. Ломоносова
Проект: «Деканат+», «Schedule-bot»

Примеры применения СПО в вузах

Аннотация

Применение СПО в сфере высших учебных заведений может принести для них огромную пользу. Из учащихся, преподавателей и прочих сотрудников университетов может сформироваться довольно крупное и серьезное сообщество, создающее и дорабатывающее приложения, автоматизирующие процессы, происходящие внутри университетов.

Как пример можно рассмотреть два приложения, действующих на физическом факультете ПГУ им. М. В. Ломоносова.

Первое — jabber-бот предназначенный для информирования студентов, преподавателей и простых сотрудников университета об интересующих их событиях. На данном этапе бот имеет возможность получать информацию о расписании с сайта отдельного факультета, сохранять ее и выдавать по запросу пользователя.

Также возможна индивидуальная настройка бота под нужды пользователей. К примеру, установка автоматического оповещения в заданное время и/или с заданной регулярностью. Или выдача информации

по сокращенному запросу (не надо постоянно указывать курс, группу, факультет).

Второе — АРМ для диспетчера деканата. Оно позволяет автоматизировать часть задач сотрудника деканата, связанных с обработкой данных студентов. Также она позволяет хранить личные дела студентов, информацию об академических группах, специальностях, предметах и прочую необходимую информацию в электронном виде. Это заметно облегчает поиск необходимых данных.

Университет — это большая и сложная структура, внутри которой происходит обмен колоссальными объёмами информации. Он представляет огромный интерес в плане автоматизации отдельных процессов.

Посредством различных крупных ИС или небольших АРМ можно в разы упростить повседневные задачи большинства сотрудников или же облегчить жизнь студентам, предоставив им различную полезную информацию в более доступном и понятном виде. А последующая интеграция созданных сервисов может принести еще больше пользы.

Так почему же не воспользоваться концепцией СПО при разработке подобных приложений? Почему бы не доверить их разработку сообществу, заинтересованному в итоговом результате, делающем, в первую очередь, для себя. Таких людей должно найтись не мало, поскольку практически в любом крупном вузе имеются факультеты, выпускающие специалистов, в сфере информационных технологий.

В качестве примера можно привести 2 небольших открытых системы, действующих в рамках ПГУ, точнее, на текущей стадии — в рамках физического факультета ПГУ. Первая из них — «Schedule-bot», информационный джаббер-бот, который обеспечивает студентам простой и удобный доступ к информации, связанной с процессом обучения, например, к расписанию. Вторая — «Деканат+», АРМ для сотрудника деканата, работающего со студентами. Эта система позволяет упростить процесс выдачи необходимых студентам справок, документов; она позволяет хранить в электронном виде данные о результатах сессий и прочую полезную информацию.

Schedule-bot — это бот, написанный для работы по протоколу jabber (xmpp), предоставляющий информацию об университете и отдельных аспектах учебного процесса всем желающим. Протокол xmpp позволяет через транспорты связываться с учетными записями других протоколов, таких как ICQ, Mail.Agent, Vkontakte и т.п. Это

очень сильно расширит аудиторию людей, пользующихся этим ботом. Schedule-bot полностью написан на языке Python, что позволяет довольно легко формировать модульную структуру: можно без проблем менять типы баз данных, в которых хранится информация, добавлять новые возможности, новые запросы, добавлять скрипты для извлечения информации из различных источников.

Также это предоставляет широкие возможности по обновлению данных, хранимых в базе: можно сделать модуль, автоматически извлекающий данные из каких-либо источников, или модуль, дающий возможность заносить данные вручную, если их неоткуда больше взять.

Деканат+ — это небольшое автоматизированное рабочее место для работника деканата. Оно полностью написано на Qt с применением генератора отчетов eXaro, а в качестве базы данных используется sqlite3. Главной задачей этого АРМ является автоматизация основных действий, которые должен выполнять диспетчер деканата. Это такие действия, как, например, печать различных документов по просьбе студентов или кого-то из преподавателей, выставление отметок по результатам сессий, государственных экзаменов, защит курсовых и дипломов. Также система призвана хранить личные карточки студентов, данные академических групп и прочую полезную информацию.

Она может облегчить жизнь как работникам деканата, так и студентам и преподавателям, поскольку на работе диспетчера деканата завязано огромное количество процессов, проходящих на факультете.

Алексей Авдеев
Санкт-Петербург, Альт Линукс

Проект: Sisyphus, Zabbix http://sisyphus.ru, http://www.zabbix.com

Zabbix. Учёт данных дочерних нод при иерархическом распределённом мониторинге

Аннотация

Описание методики создания систем иерархического распределённого мониторинга с учётом данных дочерних нод на мастер-нодах.

Предистория

Есть организация с достаточно с большим количеством филиалов, организованных в иерархическую структуру:

- головная организация;
- региональные организации — около 80 шт. (по одной в каждом субъекте Российской Федерации);
- Местные организации — порядка 1000.

Необходимо обеспечить мониторинг узлов сети передачи данных (их доступность и состояние).

Из условия задачи ясно, что система мониторинга должна быть:

1. распределённой;
2. иерархической;
3. масштабируемой до 1000 наблюдаемых хостов (как минимум).

После предварительной проработки, в качестве средства решения задачи был выбран zabbix [1]. Структура системы принята следующей:

1. Мастер-узел (мастер-нода) в головной организации. Есть возможность наблюдения и управления всей системой из данной точки.
2. На региональном уровне (в региональных организациях) — дочерние узлы (дочерние ноды). У регионального IT-подразделения есть возможность наблюдения и управления своим узлом и подчинёнными ему системами местных организаций.
3. На местном уровне используются только агенты и прокси [2]. Возможностей наблюдения и/или управления на данном уровне нет: если они потребуются — можно расширить систему, добавив ноды следующего уровня.
4. Для отображения ситуации принята следующая парадигма: *Все важные проблемы на узле должны отображаться и на вышестоящих узлах* (пороги могут быть разные). (Этот пункт один из определяющих — требование заказчика.)

Для обкатки решения, тренировки на кошках и отлова грабель была создана модель требуемой системы, в составе:

1. мастер-ноды (изображает узел в головной организации);
2. пары дочерних нод (региональный уровень);
3. имитатор местных узлов (на данном этапе достаточно ip).

Гладко было на бумаге

В процессе работы с моделью оказалось, что в zabbix 1.8.[0-2]:

1. Все содержимое базы данных (БД) дочерней ноды передаётся в БД мастера: фильтрация отсутствует как класс. При задании требований к оборудованию это придётся учитывать.
2. Хотя данные собранные дочерними нодами и присутствуют в БД мастера (см. предыдущий пункт) — их нельзя использовать в триггерах мастер-ноды [3]. (Исследования показали, что без изменений в структуре БД это не исправить.)
3. В вычисляемых элементах данных [4] данные дочерних нод тоже использовать нельзя. (А это уже можно исправить, см. ZBXNEXT-451 [5].)
4. Zabbix-сервер не может выполнять функции прокси: нода может обработать данные, переданные прокси (это *штатный* режим), переданные же другой нодой — нет (см. пп. 1-3).
5. Несмотря на пп. 2–4, состояние дочерних нод всё-таки *можно* отображать на карте!
6. Zabbix-прокси может находится только между нодой и наблюдаемым хостом. Более сложные структуры, такие как: цепочки из прокси и взаимодействие нод через прокси — не поддерживаются [6]. (Тоже простым образом не исправляется, судя по коду.)
7. Zabbix-прокси может взаимодействовать только с одним сервером.
8. Zabbix-агент в режиме демона [7] может обслуживать несколько серверов: один в режиме активных проверок и с помощью пассивных проверок остальные.

Основные проблемы — пп. 2–4, т. к. как минимум на мастер-ноде надо каким-то образом отображать консолидированную информацию, объединяя данные самой ноды (доступность региональных узлов определяет мастер) с данными дочерних нод (состояния региональных узлов и доступность местных хостов). Пп. 6–8 — результат попыток обхода данной проблемы.

«Быстрый» вариант

Обход проблемы по «быстрому» варианту: Опрос состояния местных хостов вести не только нодой регионального уровня, но и специально обученным агентом [8]. Агента опрашивать не только дочерней нодой, но и мастером.

Плюсы:
- Используются стандартные механизмы. Патчить сервер не требуется.
- Оно работает.

Минусы:
- Уменьшается возможность манёвра: лишаемся возможности разместить ноду на местном уровне.
- Лишняя нагрузка на мастер: помимо дочерних нод приходится опрашивать и расположенных там же агентов.
- Дублирование части функционала сервера своими скриптами.
- Усложнение конфигурирования и развёртывания системы.

«Правильный» вариант

Более «правильный» вариант — научить сервер обращаться к данным дочерних нод. И наиболее очевидный кандидат на доработку — механизм обработки вычисляемых элементов данных, т. к.:
- Код обработчика достаточно компактен.
- Вычисляемое выражение хранится только в одном месте (в поле *publickey* таблицы *items*).
- Пользовательский интерфейс проверку выражения не производит (этим занимается сам сервер).

Всё это позволило расширить синтаксис обращения к ключу (первый параметр используемой функции) с `[<имя_хоста:]<ключ>`) до `[[<имя_ноды>:]<имя_хоста>:]<ключ>`.

Результат работы опубликован в репозитории Sisyphus в виде пакета zabbix-1.8.2-alt1.svn.11296.1 [9] и предложен апстриму (см. ZBXNEXT-451 [5]).

Литература

[1] Сайт проекта Zabbix. `http://www.zabbix.com`

[2] Процессы Zabbix. `http://www.zabbix.com/documentation/ru/1.8/manual/processes/`

[3] Распределенный мониторинг: данные и actions. `http://www.zabbix.com/forum/showthread.php?p=59651`

[4] Элементы данных. `http://www.zabbix.com/documentation/ru/1.8/manual/config/items`

[5] ZBXNEXT-451: Unable to use data item from a subsidiary of nodes on the master node.
https://support.zabbix.com/browse/ZBXNEXT-451

[6] Chain Proxy.
http://www.zabbix.com/forum/showthread.php?t=14337

[7] Zabbix агент (UNIX, Standalone демон). http://www.zabbix.com/documentation/ru/1.8/manual/processes/zabbix_agentd

[8] Расширенные Zabbix агенты. http://www.zabbix.com/documentation/ru/1.8/manual/tutorials/extending_agent

[9] I: Zabbix: Патч для обращения к данным дочерний ноды.
http://solo-oboroten.livejournal.com/91758.html

Федор Зуев
Иркутск, ИЗК СО РАН

Экономика творчества, экономика копирайта, экономика свободного софта

Аннотация

Продемонстрированы некоторые подходы к построению экономической теории творческой деятельности, исходя из ее отношений с общеэкономическим развитием. Показано, что (в экономически значимых масштабах) творческие профессии являются следствием и функцией расширенного воспроизводства, и прослежены следствия этого тезиса для экономики, основанной на копирайте, и экономики свободного софта как частных случаях. Теория сформулирована в терминах классической (марксовой) политэкономии, но, вероятно, может быть легко переформулирована в терминах альтернативных экономических теорий.

Экономика творчества

Излагаемые ниже соображения сформулированы в терминах классической (марксовой) политэкономии. Вероятно, значительную их часть можно переформулировать в терминах других экономических теорий, однако я этим не занимался и не собираюсь.

Во первых, какова экономическая природа творчества и его результатов, от стихов до научных открытий? Уподобление изобретений и сочинений товарам, на что-то большее чем художественная метафора претендовать не может, во всяком случае, если мы отличаем творческую работу по созданию новых сущностей от физической работы по изготовлению экземпляров этих сущностей. Экономически наиболее важной чертой любого творчества является совершаемое им приращение производительных сил.

Производительные силы — совокупность инструментов, дающих человечеству контроль над Природой. Среди производительных сил имеются материальные (налаженные транспорт и связь), экономические (концентрация капитала, кредиты), организационные (конвейер, разделение труда) но нас сейчас интересуют те из них, которые сводятся к знаниям и умениям. Как увеличивают производительные силы изобретения или прикладные программы — понятно. В крайнем случае даже приключенческий роман или компьютерная игра могут быть представлены как увеличение — тривиальное — производительных сил на возможность изготовления соответствующей книжки или компьютерной игры.

Также замечаем, что рост производительных сил — не только результат, но и условие всякой творческой деятельности. Появление таковой в обществе в сколь-нибудь экономически значимых масштабах требует наличия расширенного воспроизводства, при простом воспроизводстве оно невозможно.

Простое воспроизводство в политэкономии — модель, при которой общество потребляет на свое воспроизводство (включая «производство» людей) все, что производит, в противоположность расширенному воспроизводству, при котором между производством и потреблением имеется постоянный положительный баланс. Творческая деятельность финансируется из этого баланса. Процент прибавочного продукта, который идет на финансирование творчества, определяется культурными, политическими, экономическими обстоятельствами.

Отсюда устанавливаем важное свойство творческой экономики — деньги, затрачиваемые в каждый конкретный момент как на творческую деятельность в целом, так и на ту или иную конкретную ее область, определяются внешними по отношению к этой деятельности обстоятельствами, и изнутри этой области изменены быть не могут. Это относится как к обществу в целом, так и к каждому отдельному потребителю. Если в доходе общества выпадает миллион рублей

на станковую живопись, то сколько бы художников ни было и как бы они ни организовывались промеж себя, они получат этот миллион рублей, не больше и не меньше. Если потребитель Вася имеет в своем бюджете сто рублей в месяц на покупку книжек — то истратит он сто рублей, независимо от того, придется на эти сто рублей десять книжек, одна, или одна в год. С другой стороны, творческий труд может влиять и часто влияет на величину прибавочного продукта на следующем цикле воспроизводства и таким образом — на величину собственного финансирования в будущем.

Указывали, что может существовать субституция, взаимозаменяемость для потребителя как между различными видами творчества, так и между результатом творческого и нетворческого труда. Даже если учитывать это обстоятельство, маневр субституции для софта невелик.

Экономика копирайта

Копирайт, патенты — установленное государством исключительное право (иначе говоря — монополия) на определенные производительные силы. Ограничимся рассмотрением монополии на изготовление экземпляров софта, в остальных случаях рассуждения аналогичны.

Правообладатель имеет возможность определять как цену, так и количество выпущенных экземпляров. Однако общий валовый доход фиксирован и находится вне его контроля, как показано выше. Представление о том, что задирая цены можно увеличить общий доход — неверны. Что действительно происходит, так это уменьшение рынка и как следствие — вытеснение конкурентов, предлагающих субституты. Замечаем, что рынок проприетарного софта, — один из самых монополизированных.

Представление о том, что исключительное право увеличивает суммарные платежи потребителей, — на самом деле довольно поздняя идея (конец XX — начало XXI века). До середины XX века исключительные права рассматривались прежде всего как способ перераспределить доходы среди производителей, давая среди них преимущества авторам и стимулируя таким образом сочинительство — то есть классический размен потребительных трат на капиталовложения.

Учитывая, что стоимость собственно изготовления экземпляров софта ненулевая, можно показать, что в пределе правообладателю

выгодно выпускать как можно меньше экземпляров по как можно большей цене. Эта тенденция, однако, перебивается неэластичностью спроса.

Копирайт, как и любая монополия, влечет сверхприбыли. Неверным однако будет рассчитывать, что вся величина этой сверхприбыли оседает в единоличный карман правообладателя. На самом деле, как и в других случаях когда сверхприбыль носит систематический, массовый и предсказуемый характер, прибыль размазывается вниз по всей производственной пирамиде. Цены на товары и услуги для правообладателей вздуваются пропорционально величине сверхприбылей, следом вздуваются цены на товары и услуги для производителей товаров для правообладателей и т. д., возникает множество издержек, излишних с чисто производственной точки зрения, но необходимых, чтобы удержатся на копирайт-рынке, возрастают риски. Как результат, нормы прибыли уравниваются до общего уровня.

Экономика и свободный софт

Не буду здесь обсуждать конкретные бизнес-модели свободного софта, а ограничусь только некоторыми общими обстоятельствами.

Рассмотрим случай, когда свободный софт полностью вытесняет проприетарные программы из какой-то области. В чистом виде такого пока нет нигде, но существует ряд областей, где ситуация близка, например MTA, компиляторы, браузеры, текстовые редакторы. В этом случае, по идее, свободный софт должен получить весь тот бюджет, которым ранее располагали проприетарные издатели. Количественных измерений такого рода не проводилось, но на глаз текущая сумма значительно меньше. Это может обьясняться разными причинами, например тем, что эти области вообще давно пришли к насыщению, или тем, что реальное развитие в этих областях (сейчас сосредоточенное в немногих центрах) на круг обходится дороже, чем многократная имплементация одной и той же функциональности, или неучетом каких-то категорий расходов (например, чисто исследовательских). Иными словами, требуется дальнейшее изучение.

Более типична ситуация, когда свободные программы конкурируют с проприетарными. В данной ситуации оптимальная стратегия пользователя в том, чтобы пользоваться по возможности свободным софтом, но платить — при невозможности решить задачу наличными средствами — в первую очередь за проприетарные (поскольку оплата

развития свободных программ требует или большого единовременного платежа, или сложной организации). Таким образом, свободный софт проигрывает проприетарному в краткосрочной перспективе, но теснит в долгосрочной.

Свободному софту, однако, позволяет поддерживать баланс то преимущество, что он свободен тех накрученных накладных расходов, которые диктует проприетарному софту экономика копирайта (см выше).

Анатолий Якушин
Москва, ALT Linux

Свобода подлинная и мнимая (Пути коммерциализации свободного ПО)

Аннотация

В докладе рассматриваются основные критерии свободного программного обеспечения в трактовке фонда свободного программного обеспечения, доказывается недостаточность данных критериев для практического использования, рассматриваются возможные пути коммерциализации СПО.

Современный рынок программного обеспечения не отличается высокой степенью зрелости. Данная ситуация обусловлена в первую очередь новизной программного обеспечения как товара. Существуя менее пятидесяти лет, рынок ПО быстро преодолел стадию первичной капитализации в восьмидесятые годы двадцатого века и пришел к нынешнему стойкому состоянию, которое отличает несовершенная конкуренция и олигополия на большинство видов продуктов и услуг. [1]

Наличие несовершенной конкуренции усиливают существующие барьеры для вступления в отрасль в виде правовых ограничений, высоких издержек вступления и дифференциации продукции.

На общем фоне экономического состояния отрасли создание успешных коммерческих продуктов под свободными лицензиями, а также коммерциализация ранее разработанного СПО является весьма проблематичной. Небольшой перечень успешных компаний, разрабатывающих коммерческое СПО и невысокая их доля в суммарных

доходах отрасли требует изучения с целью выработки общих моделей поведения на рынке.

Первой проблемой, с которой сталкиваются коллективы разработчиков, пытающиеся выйти на рынок с коммерческим СПО, является противопоставление свободного программного обеспечения принципам коммерции, как таковым. Широко распространено заблуждение, что СПО по своей сути не может быть коммерческим. Однако данный постулат является в корне неверным и проистекает от смещения понятий.

Следует понимать, что с ростом отрасли ИКТ значительное развитие получил феномен разработки и оборота массового коммерческого программного обеспечения.

Коммерческое программное обеспечение — это программное обеспечение, разрабатываемое и поддерживаемое в рамках соответствующей специализации бизнеса на заказ для конкретного конечного пользователя, либо с расчетом на заранее не определенный круг конечных пользователей с целью извлечения прибыли. [2]

Коммерческое программное обеспечение следует отличать от так называемой «домашней» разработки программ (разработки силой самой организации — конечного пользователя) и от разработки программ вне коммерческого контекста вообще (в ходе научного экспериментирования, учебных проектов, как хобби и т. п.).

После того как компьютерные программы были признаны объектом авторского права (в том числе, исключительных имущественных прав), в отрасли оформились два основных подхода к разработке и использованию программного обеспечения — свободное программное обеспечение (СПО) и собственническое (проприетарное) программное обеспечение.

С целью исключения терминологической путаницы, следует различать понятия «коммерческое», «некоммерческое», «свободное» и «проприетарное» программное обеспечение. И свободное и проприетарное ПО может быть как коммерческим так и некоммерческим, в зависимости от первоначальных целей разработчика, при этом наличие свободных или несвободных признаков ПО никак не связаны с возможностью или невозможностью коммерческой выгоды, обе рассматриваемые модели присутствуют на рынке продаж ПО.

Дополнительный анализ условий лицензирования свободного и проприетарного программного обеспечения и изучение жизненного цикла типичных представителей этих типов программного обеспече-

ния показывает, что на сегодняшний день нет никакой принципиальной разницы в жизненном цикле программ с разными условиями лицензирования.

Несмотря на достаточно явные признаки возможности коммерциализации СПО, существующая мифологема «некоммерческого продукта» создает серьезные проблемы привлечения венчурного капитала при создании новых продуктов под свободной лицензией. Отдельные попытки решить данную проблему путем создания Open Source Definition еще более усугубили ситуацию, в первую очередь потому, что авторы данной инициативы вместо экономического анализа пытались оперировать эмоциональными категориями. [3, 4]

Основным барьером на пути инвестиций в СПО проекты является типовая модель поведения инвестора, при которой обязательным условием успешной инвестиции является наличие патентных ограничений либо некого «ноу-хау», защищенного проприетарной лицензией. [5] Использование свободной лицензии разработчиком приводит либо к прямому отказу от инвестиций, либо к неоправданному завышению рисков проекта.

Подробное изучение современных свободных программных продуктов показывает, что во многих случаях право на модификацию кода не является препятствием для сохранения эксклюзивных возможностей разработчика.

Если во времена появления свободных лицензий право на модификацию позволяло практически любому пользователю подробно изучить исходный текст в силу незначительного объема последнего и модифицировать его за счет собственного уровня квалификации, то сегодня данная свобода фактически является декларативной.

При объеме исходного текста в миллионы строк, использовании специальных сред и инфраструктур для работы с кодом наличие только текста программы без соответствующей документации и программного обеспечения является только заявлением и ничем больше. Также следует понимать, что программное обеспечение, свободно распространяемое в бинарном виде, работающее без ошибок, не требующее дополнительной настройки и документации не может быть коммерческим по умолчанию, так как не создает условий для монетизации.

Наличие «пространства свободы» от простой декларации права на модификацию до передачи заказчику полной инфраструктуры разработки позволяет использовать СПО в самых разных коммерческих целях:

- Выпуск продукта с декларируемыми правами на модификацию и получение прибыли на технической поддержке, платных обновлениях и кастомизации.
- Выпуск продукта с декларируемыми правами на модификацию и минимальным набором функций, продажа дополнительных модулей под проприетарными лицензиями.
- Выпуск продукта и передача инфраструктуры разработки крупному корпоративному заказчику или государству.
- Разовый выпуск полностью свободного продукта по предварительному заказу.
- Продажа интеллектуального капитала компании — аутсорсинг научно-исследовательских и опытно-конструкторских работ.
- Работы по системной интеграции продуктов с различными видами лицензий.

Литература

[1] Пол Э. Самуэльсон, Вильям Д. Нордхаус «Микроэкономика». 18-е издание. Издательский дом «Вильямс».
ISBN 978-5-8459-1352-4, 0-07-287207-1.

[2] М. Отставнов «Перспективы свободного программного обеспечения в сфере государственного управления и бюджетном секторе экономики».

[3] Eric Steven Raymond «The Magic Cauldron» `http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/`

[4] Christopher Browne «The Economics of Free Software». `http://linuxfinances.info/info/freeecon.html`

[5] Chesbrough, H.W. (2003). Open Innovation: The new imperative for creating and profiting from technology. Boston: Harvard Business School Press.

Сергей Знаменский
Переславль Залесский, УГП имени А. К. Айламазяна
Проект: КАИС «Ботик» http://wiki.botik.ru/IS4UGP

К новым технологиям информационной поддержки сложных проектов

Аннотация

Описываются идеи контекстно-автономной информационной системы[1], нацеленной на информационную поддержку сложных научно-образовательных проектов будущего. Система существенно базируется на свободном программном обеспечении.

Проблемы организации сотрудничества в научно-образовательных проектах

Речь пойдёт не об инновационных проектах, когда разработанные новейшие технологии понятными путями внедряются в производственные процессы. Речь об остро нуждающихся в качественной информационной поддержке фундаментальных проектах, когда устойчивы цели исследования, а методы и технологии непредсказуемо изменяются по мере переосмысления и уточнения задач. Для такой деятельности требуется создание и реорганизация подвижных иерархий подпроектов, использующих разнообразные часто изменчивые постановки экспериментов и структуры экспериментальных данных, имеющих существенные взаимосвязи. Подобные большие проекты немыслимы без непосредственного участия нескольких групп программистов.

Система информационной поддержки такой деятельности должна удовлетворять следующим требованиям:

- полная сохранность любой внесённой информации: внесённая пользователем в интерфейсах системы информация не теряется ни при каких обстоятельствах;
- сложные фоновые обработки данных с учётом приоритетов;
- возможность неограниченно гибко перекраивать систему «на ходу» без приостановки обслуживания;

[1] грант РФФИ 09-07-00470-а

- возможность сравнивать, сопоставлять и оценивать работу, проделанную в системе различными пользователями, по различным показателям, как объективным (затраченное время, объёмы внесённых изменений), так и сведением внутренних и внешних экспертных оценок.

Такая постановка подразумевает высокий уровень защищённости от ошибок разработчиков-пользователей, предполагающий гибкое и прозрачное управление разграниченным доступом к данным системы и, в частности, к исполняемому коду.

Желательна при этом также высокая доступность системы: при исправной не перегруженной операционной системе и работающей сети отклик на каждый запрос пользователя должен приходить в течение малой доли секунды (при этом пользователь сможет открыть несколько окон для комфортной работы в системе).

Машина времени на базе СПО

В настоящее время принципиально не существует коммерческих технологий, позволяющих создать требуемую систему. Свободное программное обеспечение даёт необходимую свободу для успешных поисков фантастических решений новых задач, казавшихся неразрешимыми.

Решение сформулированной проблемы, реализуемое в Переславле — это машина времени с двумя параллельными мирами. Все пользователи, кроме программистов, живут в реальном мире. У программистов к нему есть ещё виртуальный мир, в котором на реальность накладывается виртуальный слой тестовых данных. Процессы, обслуживающие отладку обновлений в виртуальной реальности, пишут изменения в виртуальный слой и никак не могут повлиять на реальное информационное наполнение.

Машина времени означает возможность пользоваться информационным наполнением момента прошлого, игнорируя последующие изменения. Можно запустить прошлогодний сервис и он выдаст ровно ту же информацию, которую он выдал бы, будучи запущен в прошлом году в заданное время. Можно двигаться по шкале времени, отслеживая происходившие изменения.

Битемпоральные реляционные системы не способны поддержать работу машины времени уже потому, что задача индексирования с учётом возможности поиска с заданием момента времени по всей ви-

димости ранее не рассматривалась, как не рассматривалась и задача индексирования данных с разграниченным доступом. Однако для решения таких новых задач вполне пригодны нереляционные свободные СУБД, отличающиеся высоким качеством документации и исходного кода.

На этом пути невозможность использования готовых стандартных решений ни в архитектуре, ни в интерфейсах, создаёт головоломные сложности:

- так организовать записи и хранения всех данных, включая исполняемый код, чтобы полная воспроизводимость истории и особенно обработка поисковых запросов на любой момент времени гарантировалась в реальном времени без привлечения избыточных ресурсов;
- так организовать фоновую обработку информации, чтобы динамически определяемая приоритетность учитывала текущую заинтересованность пользователей (не считать статистику и не обновлять индексы на страницах, которыми никто сейчас не пользуется, если есть более востребованные дела);
- так реализовать пользовательские интерфейсы, чтобы пользователь видел, что, где и как меняется в системе и насколько актуальны те данные, с которыми загрузилась новая страница.

Практическая реализация

Разработка системы ведется открыто на `http://wiki.botik.ru/IS4UGP` силами студентов 1—5 курсов. Сейчас работает далёкий прототип, включающий модуль трекинга учебной и производственной практик и курсовых и дипломных проектов с процедурами утверждения/изменения темы (организации), выбора/смены руководителя и других параметров, а также автоматической оценкой регулярности и содержания периодических отчетов о проделанной работе, рабочий журнал преподавателя и подсистема поддержки научно-практических конференций с перекрестным рецензированием, анонимным согласованием мнений рецензентов и оцениванием всех форм участия. Пробная реализация ядра машины времени намечена на конец лета. Разработка ведётся в linux и ориентируется на перспективу передачи исходного кода в свободный доступ по достижению приемлемой функциональности.

Игорь Воронин
Шатура, ИПЛИТ РАН

Использование СПО для визуализации обработки результатов расчета температурных полей, полученных с использованием РСС

Аннотация

На основе СПО выполнен расчет температурного поля в металлическом стержне. Результаты расчета достаточно точно коррелируют с результатами экспериментальных исследований на основе РСС. Произведена визуализация полученных результатов.

Лазерное излучение с высокой эффективностью применяется в технологии обработки материалов для сварки, резки, термообработки и т. д. При лазерном воздействии наиболее существенными факторами, влияющими на физико-механические свойства материалов, являются значения максимальных температур и скорости нагрева и охлаждения обрабатываемого потоком высококонцентрированной энергии изделия. Для расчета температурных полей при лазерной обработке материалов используют аналитические и численные методы [1-2]. При использовании аналитических методов все случаи нагрева материалов лазерным излучением сводятся к трем упрощенным схемам, учитывающим основные особенности этого процесса: тонкая пластина, полубесконечное тело, многослойные системы. Приближенные аналитические решения дифференциальных уравнений теплопроводности могут использоваться в относительно простых схематизированных моделях для проведения качественного и приближенного количественного анализа тепловых явлений. Возрастающие возможности вычислительных средств обуславливают широкое применение численных методов решения нестационарных трехмерных задач нелинейной теплопроводности. В работах [3, 4] рассмотрены особенности постановки таких задач, а для численного анализа рекомендуется использовать метод конечных разностей. При повышении точности расчета по методике, приведенной в работе [5], объем вычислений значительно возрастает, что требует больших затрат процессорного времени. И тем не менее, таким образом мы можем получить наиболее точный расчетный результат на основе расчета суммы произведений

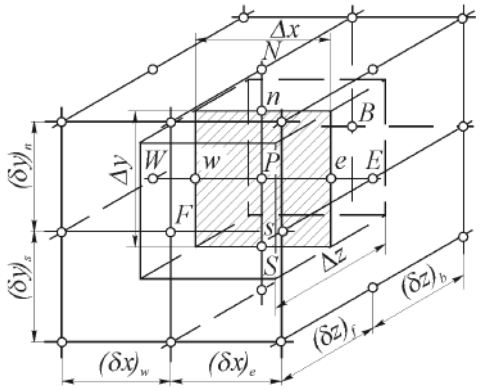

Рис. 1. Семиточечтный орт

проводимости a и температуры Tm в 6-ти точках координат (i, j, k) за предыдущий момент времени.

На рис. 1 эти шесть точек обозначены, как F, S, E, W, N, B, мы расчет производим для седьмой точки P. Для расчета используем следующий набор кода на языке С:

```
if (e<rZ-1){
  a = RaschetK(Tp.GetElement(i,j,e+1))*dy*dx/dtz;
  S1 = S1 + a*Tp.GetElement(i,j,e+1);
}
cv = RaschetCv(Tr);
Tr = (S1+dV*Scp + dV*Tr*cv/dt)/(S2+dV*cv/dt);
```

Здесь a — это проводимость (1/K)которую рассчитываем следующим образом: (i,j,k)=K(i,j,k)* dy* dz/dx(i,j,k). RaschetK(i,j,k) является коэффициентом теплопроводности на гранях контрольного объема dV, его мы берем из таблицы заранее предопределенных значений. Далее мы рассчитываем сумму произведения проводимости и температуры в шести точках, в предшествующий момент времени.

```
for (g=1; g<7; g++) // Шесть точек — шесть итераций!
{ at = at + a(i,j,k)* Tm(i,j,k);} // получаем одно число из суммы
```

Параметры инициализации расчета задаем следующие:

Рис. 2. Схематичное изображение расчитываемой детали

1. Степень дискретизации сетки берется со следующим шагом: dx, dy, dz $-->$ 0,5 $-$ 0.8 $*$ 0.001 (m).
2. дискрета времени dtm = 1 $-$ 2 $*$ 0.01 (c), число
3. dV(i,j,k) = dx$*$ dy$*$ dz (m) контрольный объем, число
4. Sc(i,j,k) — Мощность излучения ВТ, в каждой точке величина составляет 1.0e10 матрица по площади. Излучение Гаус. площадь пятна 15 мм кв. длина* ширина: 0.01$*$0.0015 (m). Cv — число — объемная теплоемкость, функция от температуры. Берется из таблицы зависимостей — материал от температуры. dV(i,j,k)$*$ (Sc(i,j,k)+ (Tm(i,j,k)$*$ Cv(Tm_i_j_k))/ dtm).

Граничные условия определяем исходя из следующих допускаемых значений:

- Определяем металл как (Ст45 или, возможно, другое):

```
float RaschetCv(float temp) if (temp<=373)
  return 2470000;
float RaschetK(float temp)
// dla tit splava if (temp<=373) return 13;
```

- Задаем массив теплопроводности (зависимость от температуры);

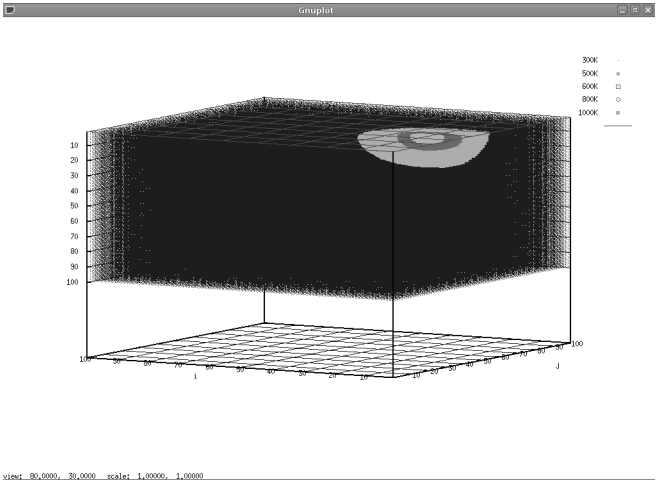

Рис. 3. Результат расчетов по прошествии времени

- Определяем размеры ячейки нашего орта как
  dx, dy, dz = 0.5e−3 // *kol−vo tochek rX = 10*;
- Определяем дискрету времени

  ```
  dt = 1.0e-2;// v sec;
  Tmax = 100;// kol-vo iteracii po vremeni
  ```

  — итого 10 секунд получается на весь процесс;
- Определяем скорость движения детали — 6 мм/сек
- Определяем суммарный поток энергии Sc=8.8e7 в одной точке. Для Гаусса в центре потока у детали интенсивность ниже — по краям — выше. 10*2.5 mm или 70*15 точек.
- Делаем предположение, шейка коленвала диаметром 35 мм. шириной 10 мм.
  - Источник эенргии имеет плоский фронт;
  - Температурное поле квазистационарное;
  - Делаем расчет во всех точках детали для решения обратной задачи термодинамики методом подстановки.

Программный код скомпилирован на gcc version 4.4.1 20090725 (ALT Linux 4.4.1-alt3) (GCC). Далее выполняем запуск расчета и получаем прикладные данные:

1. Рассчитываем проводимость — a, для точки, с учетом коэффициента теплопроводности.
2. Рассчитываем сумму для шести точек произведения проводимости и температуры.
3. Рассчитываем сумму проводимости для шести точек.
4. Рассчитываем объемное поглощение потока энергии.
5. Рассчитываем объемную теплопроводность.
6. Рассчитываем температуру в искомой точке P. Сложив и разделив предыдущие значения.
7. Записываем значение в массив (*три координаты, время, значение температуры*). Объем всех точек детали за нулевой момент времени. Выгружаем массив в файл через 2 итерации расчетов, при этом отбрасываем дробные части.
8. Отображаем срез фрагмента детали как плоский массив на нашей детали, температура 5 градаций различается — цветом. Для этого преобразуем файл с температурами в файл с точками координатами(X,y) в 5 наборов столбцов. Обрисовываем для каждого набора столбцов цвет своей точки. Синий, желтый, красный, белый. Рисуем, используя программу «gnuplot 4.2 patchlevel 6». Раз различные рисунки для каждого цвета набор точек (x,y).
9. Изменения температуры от потока энергии берутся из массива пятна энергии за следующий момент времени.

Полученные результаты изображены на рис. 3.

Контрольные параметры показаний температуры получаем при помощи сбора показаний с датчиков температуры, передающих данные по РСС.

Литература

[1] Технологические лазеры: Справочник: В 2 т. Т.1: Расчет, проектирование и эксплуатация [текст] / Под ред. Г. А. Абильсиитова — М.: Машиностроение, 1991. — 432 с.

[2] Гуреев, Д.М. Основы физики лазеров и лазерной обработки материалов. [текст] / Д. М. Гуреев, С. В. Ямщиков — Самара 2001. — 392 с.

[3] Основы лазерного термоупрочнения сплавов: Учеб. пособие для вузов [текст] / Под ред. А. Г. Григорьянца — М.: Высш. шк., 1988. — 159 с.

[4] Григорьянц, А.Г. Основы лазерной обработки материалов. [текст] / А. Г. Григорьянц — М.: Машиностроение, 1989. — 304 с.

[5] Применение метода контрольного объема для расчета температурного поля при лазерном воздействии [текст] / С. П. Мурзин, А. В. Меженин, Е. Л. Осетров — Самара: Самарский государственный аэрокосмический университет, 2008.

Евгений Сыромятников
Москва, ALT Linux

Проект: MoinMoin wiki engine `http://moinmo.in/`

MoinMoin 2.0: вики-сервер нового поколения

Аннотация

Понятие «вики» существует не первый десяток лет, и технологии, его реализующие, используются практически повсеместно. Простота оригинальной идеи во многом благоприятно сказалась на популярности данного способа совместной работы с гипертекстовой информацией. В то же время, современные вики-сервера (в особенности, наиболее популярные и функциональные из них) зачастую предоставляют множество дополнительных возможностей, которые весьма разнообразны, но в то же время разрозненны и зачастую плохо интегрированы друг с другом. Во второй версии MoinMoin его создатели пытаются пересмотреть концепцию вики в соответствии с существующими реалиями и появившимися в процессе активного использования новыми требованиями и создать на её основе вики-сервер нового поколения, по функциональности не уступающий существующим решениям.

Что вики есть

> The simplest online database that could possibly work.
>
> Ward Cunningam, `http://www.wiki.org/wiki.cgi?WhatIsWiki`

Как многим, вероятно, известно, вики — (веб-)сервис, позволяющий пользователям свободно создавать и редактировать содержимое

размещённых на вики-сайте страниц. Почти всегда вики-сервер поддерживает форматирование текста посредством вики-синтаксиса (минималистичного текстового синтаксиса наподобие ReST и Markdown; синтаксис различных вики-серверов — программного обеспечения, обеспечивающего работу вики — обычно отличается). Другой особенностью вики является простота указания ссылок и создания новых страниц. Также следует отметить такую важную особенность вики (являющуюся необходимым следствием простоты внесения правок) как наличие истории изменений у каждой страницы.

Наслоения

Простота изначальной концепции и её гибкость позволили вики найти применение в широком спектере приложений. Как следствие, исходя из специфики конкретной задачи, в те или иные вики-сервера вносилась различная дополнительная функциональность. Ниже приведены некоторые примеры подобных дополнений:

- Поддержка синтаксиса, отличного от вики-синтаксиса (HTML, DocBook, ReST, ...).
- Возможность хранения на вики файлов (как прикреплённых к страницам, так и отдельно существующих).
- Ограничение доступа. Наиболее продвинутые варианты реализации подразумевают полноразмерный ACL.
- Интеграция с различными сервисами аутентификации (LDAP, OpenID, ...)
- Различные системы расширений синтаксиса и функциональных возможностей.
- Различные способы хранения, индексирования и поиска.
- Управление пользователями (группы, права, переименование, ...).
- Различные упражнения с внешним видом — пользовательские стили, стили страниц.
- Локализация и поддержка нескольких языков.

Не всё легко ложится на оригинальную концепцию. Одним из ярких примеров инородных включений можно назвать поддержку сторонних синтаксисов и файлов — чаще всего они обрабатываются особым образом, что создаёт проблемы для авторов расширений и вообще при любых существенных изменениях в коде вики-сервера.

В результате выжеизложенного постепенно стала назревать необходимость в обновлении концепции. При этом очевидно желание сохранения её простоты и легковесности и не меньшей гибкости, чем было заложено изначально.

Концепция MoinMoin 2.0

Одним из возможных вариантов реализации некоторой новой концепции является MoinMoin 2.0 — активно разрабатываемая на данный момент новая версия вики-сервера MoinMoin. В его основе лежит ряд положений, среди которых имеются следующие:

- Унификация хранимых на вики сущностей:
  - Единый вид объекта — Item;
  - Объект имеет версионируемые данные и набор метаданных (среди которых MIME-тип) как для всего объекта, так и для отдельных версий.
- Единый способ обработки различных форматов на основе wiki DOM и конверторов в выходные форматы.
- Подсистема хранения:
  - Выделение storage backend (который может быть любым: на основе файлов, БД, DVCS...);
  - mount-like подключение различных хранилищ.

MoinMoin 2.0 содержит ряд других решений (по сравнению с веткой 1.x, история которой тянется с 2000 года в качестве форка PikiPiki, — переработка и унификация расширений, темплейтинг на основе Jinja2, поддержка совместного редактирования), но существенными в плане концепции являются именно перечисленные выше положения.

Текущее состояние

На данный момент MoinMoin 2.0 активно разрабатывается, но ввиду малого числа основных разработчиков и большого количества кода, который требуется написать, релиз планируется не ранее 2011 года. На данный момент уже реализованы основные части системы (storage, Wiki DOM, Item) и ведётся работа над функциональным наполнением (конверторы в раличные форматы, темплейтинг, различные storage backend).

Ссылки

1. MoinMoin — `http://moinmo.in/`
2. MoinMoin 2.0 — `http://moinmo.in/MoinMoin2.0`
3. Wiki DOM — `http://moinmo.in/MoinDev/WikiDom`
4. ReST (Restructured text) — `http://docutils.sourceforge.net/rst.html`

Светлана Гайворонская
Москва, МГУ ВМК

Разработка расширения протокола RFB для улучшения отзывчивости тонких клиентов

Работа посвящена изучению и исследованию средств организации удалённого рабочего места. *Удалённое рабочее место* — программно-аппаратный комплекс, в котором приложения физически выполняются не на том компьютере, с которым работает пользователь. При работе удалённого рабочего места задействованы такие аппаратные компоненты, как сервер приложений и терминал. Связь между сервером приложений и терминалом осуществляется при помощи *терминального протокола*.

В работе рассматриваются наиболее распространённые в настоящий момент средства организации удалённого рабочего места (RDP, VNC[1], NX, X Window System и Sun Ray). В результате проведённых исследований показано, что ни одно из вышеупомянутых средств не обеспечивает прозрачной работы пользователя (не отличимой от работы на локальной машине) на удалённой машине. Прежде всего, проблема возникает из-за нагрузки на канал передачи данных и больших задержек в обновлении элементов интерфейса (отзывчивости).

Работа базируется на средстве организации удалённого рабочего места VNC, использующего терминальный протокол RFB[2]. Протокол имеет большой потенциал ввиду заложенной в него возможности расширения. В рамках работы разработано расширение протокола RFB, организующее кеширование графической информации на стороне клиента VNC. Серверная часть расширения представлена ана-

лизатором данных и буфером, поддерживаемым в согласованном состоянии к кешем клиента. В качестве данных, поступающих на вход анализатору, выступают обновлённые области экрана. Для каждой из областей вычисляется идентификатор, в качестве которого выступает хэш-функция md5, подсчитывается количество раз, которое эта область встречалась, и на основе этого принимается решение о формировании различных протокольных данных. Область, встретившаяся 1 раз, передается по сети в виде растрового изображения; для области, встретившейся 2 раза, формируются протокольные данные с пометкой клиенту закешировать ее; для области, встретившейся 3 и более раз, формируются протокольные данные с пометкой клиенту взять ее из кеша.

В рамках работы было проведено исследование, как модификация обрабатываемых анализатором данных влияет на сокращение трафика, передаваемого по сети, и, соответственно, на повышение эффективности работы терминального протокола RFB. Были разработаны три различные стратегии формирования обрабатываемых данных. В качестве первой стратегии рассматривается исходная последовательность обновлённых областей (последовательность, сформированная средством VNC). В качестве второй стратегии рассматривается сеточное разбиение исходной последовательности обновлённых областей. При сеточном разбиении на каждый элемент исходной последовательности накладывается сетка с определённым шагом (в ходе экспериментов было показано, что наиболее оптимальным является шаг размером в 20 px), и новая последовательность формируется из ячеек сетки. В качестве третьей стратегии рассматривается охватывающее разбиение исходной последовательности. При охватывающем разбиении сетка (с шагом в 20 px) накладывается на дисплей, и новая последовательность формируется из тех ячеек сетки, которые имеют пересечение с элементами исходной последовательности.

Проведённые эксперименты показали, что использование разработанного расширения совместно с исходной последовательностью областей сокращает нагрузку на канал передачи данных на 24—35% в зависимости от типа запущенных пользователем приложений и типа кодирования графической информации. Использование расширения на сеточном разбиении исходной последовательности сокращает нагрузку на канал передачи данных на 34—54%, на охватывающем разбиении — на 51—83%.

Литература

[1] Virtual Network Computing from ORL, [HTML] (http://wwwrsphysse.anu.edu.au/doc/Xvnc/)

[2] Richardson T. The RFB Protocol, [PDF] (http://www.realvnc.com/docs/rfbproto.pdf)

Александра Панюкова
Москва, ALT Linux

Проект: ALT Linux Children http://children.altlinux.org

children.altlinux.org как проект, существующий на обратной связи с пользователями

Обратная связь от пользователей актуальна для всех видов разработки, открытой — тем более. Люди, пользующиеся проектом, могут быть следующих видов:

- готовые вносить значимые изменения и обладающие достаточной квалификацией — идеальный случай, который встречается, к сожалению, не настолько часто, как хотелось бы;
- готовые вносить значимые изменения, но не обладающие достаточной квалификацией по части или всем вопросам — также крайне полезная группа пользователей, однако их изменения перед добавлением в проект необходимо пропускать через премодерацию/обработку участником проекта, обладающим достаточной компетенцией/полномочиями;
- готовые сообщать об ошибках, содействовать при уточнении причин, внятно формулировать улучшения/изменения, которые им полезны — таких пользователей также меньше, чем хотелось бы;
- готовые поделиться какой-то идеей/соображением/найденной ошибкой, не тратя на это много времени;
- молча пользующиеся — молча страдающие.

Последняя и предпоследняя группы — самые большие. На практике было бы интересно их разделять, так как предпоследняя в некоторых случаях — кладезь идей, того самого, что нужно действительно рядовому пользователю.

Интересны ли проекту пользователи, которые не в состоянии пользоваться системой отслеживания ошибок? Несомненно, проекты бывают разные, но существуют и те, которые могут много терять из-за невозможности/банального нежелания написания FR или описания ошибки. Сюда можно отнести множество идей и просьб, которые были изложены в различных IM или устно, и так и не вышли за пределы этих каналов. Иногда среди таких идей встречается действительно интересные изменения.

Выхода из этой ситуации толком нет — в большинстве случаев всё упрётся в изобретение очередного велосипеда для отслеживания ошибок, который в результате будет не обязательно проще или удобнее уже существующих. Интересна попытка создания такого проекта, где возможность оставления фидбека была бы максимально «встроенной», а также являлась бы ключевой в развитии проекта. Выживаемость или невыживаемость такого проекта может говорить как об удачности созданной модели, так и об удачности идеи поддержания некоторых проектов большей частью только на фидбеке от пользователей, идеи «проект вокруг коммьюнити», а не «коммьюнити вокруг проекта».

Рассмотрим случай проекта children.altlinux.org. Несмотря на то, что проект не направлен на написание какого-либо ПО, сама модель получилась достаточно характерной и интересной. Была поставлена задача создать схему, с помощью которой можно было бы получить максимум фидбека, в том числе и от «ленивых» пользователей, но хоть сколько-нибудь интересующихся судьбой проекта.

Какие составные части—уровни проекта имеются:

- Собственно текст-сюжет как некое общее направление движения. Представляет из себя набор историй (связанный общими персонажами, часто — сюжетом), в которые завернуты описания каких-то примеров взаимодействия с ПО (более точных «рецептов» и не только), принципов работы и разработки, удобного взаимодействия с другими и т. п.
- Вспомогательные описания — например, описания персонажей, мест действия и т. п. Нужно в первую очередь как ориентир для непротиворечивости создаваемых образов.
- Фидбэк от пользователей (в данном случае в эту категорию может попасть и основная команда проекта) — заметки на тему «про что ещё хотелось бы рассказать», уточнение персонажей,

предложения по радикальному изменению сюжетной линии, вырванные из контекста диалоги, продолжения и т. п.

Естественно, не все составляющие проекта с точки зрения участия интересны каждому. Написание текста-сюжета, задающего общее направление сюжета, требует хотя бы способности литературно излагать мысли, принимая во внимание уровень ниже — вспомогательные описания.

Уровень описаний — более простой с точки зрения написания текста, тут достаточно умения внятно выражать свои мысли, а также изложенное уровнем ниже. Тем не менее, это всё равно довольно трудоемкое действие, заинтересовать которым не так просто.

Уровень фидбека от пользователей — наиболее простой с точки зрения порога вхождения. Предполагается, что сюда будут попадать сообщения вроде «пусть у Маши будет красный бант» или «можно грабить караваны» и т. п.

Разница между уровнями в данном случае несколько похожа на разницу в прилагаемых усилиях, чтобы написать статью, запись в блог или запись в твиттер.

Такая структура сама по себе предполагает навигацию по создаваемым материалам — возможно, вне зависимости от вида фидбека, что достигается использованием тегов.

Реализацию можно посмотреть на `http://children.altlinux.org`. Запускать (раскручивать и созывать детей) планируется ближе к началу учебного года из-за сезонных особенностей, тогда же можно будет судить об успешности. Но уже сейчас все желающие и заинтересованные могут принять участие в проекте.

Владимир Гусев
Москва

Проект: Antique `http://www.altlinux.org/Antique`

Antique — дистрибутив для старых компьютеров

Аннотация

Доклад посвящен проекту Antique — разработке одноименного дистрибутива для старых компьютеров с использованием современной пакетной базы ALT Linux.

Идея

Идея, конечно же, отнюдь не нова. Существует приличное количество известных и весьма уважаемых т. н. лёгких дистрибутивов. ALT Linux выпускал и выпускает широкую линейку дистрибутивов разного назначения, однако первые опыты по созданию «лёгкого» дистрибутива осуществились относительно недавно. При этом подбор графической среды и приложений оказался неоднозначным — работа приложений в XFCE оказалась не столь быстрой по субъективным ощущениям по сравнению с той же KDE 3.5. Во время тестирования ALT Linux 4.0 Lite beta выяснилось, что смена XFCE на Icewm более чем в 2 раза (50—60 мб против 20—25 мб) уменьшает количество памяти, занятой системой сразу после загрузки.

Однако, как правило, одной экономии оперативной памяти мало для получения комфортной работы Linux на старом компьютере. Необходимо ощущение т. н. «лёгкости» работы приложений...

Лёгкость, lightweight — что это? Истинная ли «лёгкость», а точнее говоря — беспристрастные показатели, «циферки»? Или же лёгкость — это нечто эфемерное, кажущееся, субъективное? Эпитеты пользователей вроде «летает», «мгновенно запускается», «быстрая прорисовка» и т. д. и т. п. описывают наши ощущения, возникающие при работе в системе. К этому и нужно стремиться.

Вполне естественным решением будет по возможности сочетать эти 2 типа «лёгкости» — важным является выбор правильного ядра. Тщательный продуманный отбор программ, использующих, по возможности, нересурсоёмкие библиотеки для отрисовки интерфейса, где-то применять «стероиды» путем использования различных ухищрений (метод сборки *as-needed*, утилиты *prelink, preload*) для создания субъективного ощущения быстрой работы.

Что считать «старым» компьютером в наше время?

Старые компьютеры (или ноутбуки) можно классифицировать следующим образом:

A мало что могут — CPU 166—266 мгц, RAM 32—64 мб видео 1—2 мб;
B что-то могут — CPU 266—533 мгц, RAM 64—128 мб видео 2—4 мб;
C могут многое — CPU 533—700 мгц, RAM 128—256 мб видео 4—8 мб;
D могут практически всё — CPU 700 мгц и выше, RAM 256—512 мб, видео 8 мб и выше.

Понятие «могут-не могут» в этой классификации применимо к **современным** программам.

Понятно, что Compact 2.3 или Master 2.2 немного из другой весовой категории. Поэтому для разных категорий (задач) можно подбирать программы по 2—3 аналога — консольный (в первую очередь для категорий A и B), графический урезанный но лёгкий (A,B,C), графический полноценный и менее лёгкий (в первую очередь для D).

Почему Antique? Ведь есть ALM 2.0, 2.2, ALC 2.3 и т. д.

Так то оно так, однако даже **по сравнению с ALC 3.0 количество интересных прикладных программ значительно увеличилось**. Растет объём поддержки различной периферии, полноценно работающей и с морально устаревшими компьютерами. Этих аргументов вполне достаточно для того, чтобы сделать выбор в пользу современной пакетной базы.

Однако в этом случае приходится идти на определённые жертвы — с ростом версии ядра, Xorg, интерфейсных библиотек растет и ресурсоёмкость дистрибутива в целом. Именно поэтому хочется ещё раз отметить важность выбора наиболее удачной версии ядра 2.6.*.

Для кого предназначен Antique?

В первую очередь дистрибутив может быть интересен для для обладателей старых компьютеров (ноутбуков), желающих вдохнуть в эти машинки «новую жизнь», а также для пользователей, впервые сталкивающихся с ОС Linux, но не рискующих устанавливать незнакомую ОС на свой основной компьютер.

Предполагается, что Antique должен будет обеспечить возможность полноценной поддержки современной компьютерной периферии, безопасной работы в сети Интернет и решение на старом компьютере ряд стандартных «бытовых» домашних задач, как то:

- Создание и просмотр различных документов;
- Чтение электронных книг;
- Работа с электронной почтой;
- Мультипротокольная поддержка сервиса мгновенных сообщений;
- IP-телефония;
- Просмотр веб-сайтов;

- Прослушивание музыки;
- Просмотр видео (*зависит от конфигурации компьютера — чудо если и случается в жизни, то не в этой области. Многое зависит от формата, степени сжатия файла и т.д.*).
- Работа с фотографиями;
- Настольные игры;
- Возможность подключения и синхронизации с мобильными устройствами;
- Возможность запуска и полноценной работы с многочисленными нересурсоёмкими и весьма эффективными (даже в наше время) DOS-приложениями, коих великое множество.

Не исключено, что Antique могут заинтересоваться любители «разумного минимализма» — обладатели мощных компьютеров. А также государственные учреждения, особенно в не самых благополучных с финансовой точки зрения регионах РФ, до которых ещё не скоро дойдет обещанная смена парка устаревших машин.

Однако в первую очередь Antique — это дистрибутив для старых компьютеров. Чаще всего это лёгкие и вполне качественные японские ноутбуки семейства P-III, которых ещё очень и очень много на российском рынке компьютерного железа.

Поддержка оборудования

Повторюсь — особое внимание следует уделить выбору версии ядра. Стоит также отметить, что в последнее время наметилась тенденция к постепенному «вымыванию» поддержки старого оборудования, поэтому необходимо уделить внимание ISA, старым IDE-контроллерам и т. д.

Подбор приложений

Анализируя состав приложений, включённых по умолчанию в те или иные популярные операционные системы, можно прийти к выводу о том, что разработчики не стремятся охватить **абсолютно весь спектр задач**, так или иначе решаемых с помощью рабочей станции. По каким-то критериям (опросы, маркетинговые и прочие исследования) они отбирают несколько главных, на их взгляд, направлений,

однако это, скупо отобранное, ПО отлажено и работает «как часы» (в первую очередь это касается MacOS X).

В нашем случае, думаю, следует поступить именно так. Не нужно пытаться запихивать всё подряд... А выбранные программы следует предварительно настроить с целью обеспечить органичную программную среду с приложениями, не конфликтующими, а дополняющими друг друга.

Исходя из конфигурации конкретного компьютера следует по умолчанию предлагать тот или иной специально подобранный набор ПО. На данный момент существуют два варианта установки — base и disk (или скорее full=base+disk). Набор base предназначен для пункта A по классификации старых компьютеров (см. выше), более-менее охватывающий основные задачи, упомянутые выше. Установив дополнение disk, пользователь получает в распоряжение полный список ПО. Возможность установки дополнения disk может быть предложена во время установки ОС либо после установки при первой загрузке системы (с обязательным упоминанием необходимого минимума системных требований для нормальной работы приложений из disk).

В качестве оконного менеджера для графической среды были выбраны WindowMaker и Icewm. Грамотный подбор вспомогательных программ позволит сделать работу пользователя не менее удобной, чем с использованием оконных сред (т. н. DE — KDE, Gnome).

Дизайн Antique

Отдельный момент — эстетика, ощущение однородности, органичности софта, содержащегося в системе. Я не люблю половинчатости — если делать, так делать нормально и до конца — и причисляю себя к тем, которым нужны «и шашечки, и ехать».

Вполне понятно, что изначально создается обычный «зоопарк» программ, написанных с использованием разных интерфейсных библиотек со своими виджетами. По возможности, подбор ПО будет осуществляться по признаку «все на gtk2», или «все на qt3», и т. д. Большинство программ, которые были предложены мной, написаны на gtk2, однако есть и другие. Поэтому нужно озаботиться выбором темы виджетов, набора пиктограмм и т. д. Ну и не будем забывать о некоей уникальности внешнего вида, подбора цветовой гаммы...

Поддержка сообщества и ALT Linux Team

Хочется надеяться, что после доклада к проекту Antique подтянутся заинтересованные грамотные пользователи Linux из Community, а также хотя бы малая часть ALT Linux Team, без помощи которой проект не сможет выйти на определённый уровень качества.

Алексей Мичурин
Москва, Вымпелком («Билайн»)

Проект: Scato `http://scato.googlecode.com/`

Scato: черепаший язык для обучения навыкам программирования

Аннотация

Scato — развитый черепаший язык (язык для создания рисунков посредством программирования действий воображаемой черепашки). Язык позволяет продемонстрировать основные доктрины императивного программирования: ветвления, циклы, подпрограммы, контексты. Язык разрабатывался таким образом, чтобы, с одной стороны, быть максимально простым, а с другой стороны, подготовить ученика к изучению Pascal (Pascal используется в ЕГЭ). Среда исполнения языка не только визуализирует результат работы программы, но и позволяет выполнять программу пошагово, просматривать текущие значения переменных и параметры черепашки.

Область применения языка

Изначально Scato (**sca**lable **to**rtoise) разрабатывался как средство построения итерационных фракталов, итерационных систем и различных сложных кривых. Но простота языка привлекла к нему внимание школьных учителей информатики.

Переломный момент наступил два года назад, когда на Scato обратили внимание в школе № 30 Йошкар-Олы. Преподаватели и руководство этой школы активно используют Scato, докладывают о своём опыте на конференциях [1, 2, 3], разрабатывают методические материалы [4].

Благодаря их замечаниям и дополнениям Scato приобрёл достаточно зрелый вид и был размещён на Googlecode под лицензией BSD.

Сейчас Scato используется многими преподавателями информатики и математики.

На чём написан и где может запускаться

Scato написан на языке Python[1]. Для установки используется штатный механизм Python. Для организации графического интерфейса используется кросс-платформенная библиотека Tk[2]. Поэтому Scato может использоваться на любых версиях UNIX и Windows.

Возможности языка

Синтаксис Scato предельно прост (S-грамматика). Программа состоит из выражений, а смысл и длина выражения (количество токенов) полностью определяется первым словом. Это единственное правило полностью определяет синтаксис. Таким образом, без ущерба для языка удаётся избавиться от необходимости использования множественных скобок, точек с запятой и прочих синтаксических конструкций, которые вызывают затруднения на начальных этапах освоения языков программирования.

Зарезервированные слова максимально приближены к Pascal.

Язык содержит *средства управления черепашкой*: можно рисовать, передвигаться без рисования, гибко управлять цветом (в том числе смешивать цвета) и толщиной линии. С этих простейших команд можно начать изучение языка. На этом этапе уже можно создавать несложные рисунки, не используя никаких других средств языка.

Scato предоставляет развитые средства *преобразования системы координат*. Это не только повороты, сдвиги и масштабирования, но и произвольные аффинные преобразования.

Поддерживаются *переменные* и работа с ними: присвоения, математические операции.

Язык включает стандартный набор средств *управления ходом выполнения*: условные переходы, циклы. Причём, имеется специальная

[1] `http://www.python.org/`; BSD-подобная лицензия.

[2] `http://www.tcl.tk`; BSD-подобная лицензия.

конструкция, реализующая упрощённый цикл, с указанием только количества повторений. Это позволяет объяснять концепцию циклов на простейших примерах.

Поддерживаются *именованные процедуры*.

Поддерживаются *контексты*: локальные переменные и локализованные преобразования координат (т. е. изменения значений переменных и/или системы координат теряют свою силу за пределами области локализации).

Возможности среды исполнения

Среда исполнения имеет стандартный *набор средств работы с файлами*: загрузка программы, сохранение полученного изображения. Кроме того, вы можете включить режим «наблюдения за файлом программы»; если файл изменится, то среда автоматически выполнит новую программу.

Имеется набор средств для *управления ходом выполнения*: «остановить», «продолжить», «пошаговый режим», «просмотр значений переменных».

Scato оснащён *системой помощи*, включающей небольшую, но исчерпывающую памятку и обширную коллекцию примеров. Примеры разбиты на две группы. Первая состоит из простейших примеров, демонстрирующих отдельные возможности Scato. Эти примеры предназначены для начального изучения. Вторая группа — более сложные примеры. Здесь вы найдёте многие фракталы, итерационные системы, кривые Коха, Пеано и другие забавные и любопытные геометрические объекты.

С описанием языка и примерами можно ознакомиться в wiki на сайте проекта.

Дополнительные возможности

Scato изначально разрабатывался модульным. Все его части можно использовать по-отдельности. Поэтому вы можете легко развивать Scato, добавляя в него новые возможности. Вы также можете использовать отдельные части Scato для создания собственных приложений.

Это может быть полезным при изучении языка Python.

В комплект поставки Scato входит несколько примеров с комментариями.

Литература

[1] *Цвирко Валерий Эдуардович*, материалы конференции «VI Всероссийская научно-практическая конференция “Применение информационно-коммуникационных технологий в образовании” (“ИТО-Марий Эл-2009”)», доклад «Опыт внедрения свободного программного обеспечения в общеобразовательной школе». (`http://iso.marsu.ru/conference/ito2009/`)

[2] *Пинешкин Сергей Павлович*, материалы конференции «VI Всероссийская научно-практическая конференция “Применение информационно-коммуникационных технологий в образовании” (“ИТО-Марий Эл-2009”)», доклад «Использование системы SCATO для обучения основам программирования» (`http://iso.marsu.ru/conference/ito2009/`; текст тезисов: `http://ito.edu.ru/2009/MariyEl/IV/IV-0-8.html`)

[3] *Пинешкин Сергей Павлович, Чугунова Лилия Александровна*, материалы конференции «VI Всероссийская научно-практическая конференция “Применение информационно-коммуникационных технологий в образовании” (“ИТО-Марий Эл-2009”)», доклад «Из опыта использования ПСПО на уроках информатики в МОУ средняя общеобразовательная школа 30 г. Йошкар-Олы» (`http://iso.marsu.ru/conference/ito2009/`; текст тезисов: `http://ito.edu.ru/2009/MariyEl/IV/IV-0-9.html`)

[4] *Чугунова Лилия Александровна*, методист сообщества «Информатика и ИКТ» проекта «Открытый класс». (`http://www.openclass.ru/`, `http://www.openclass.ru/weblinks/29042`)

Дмитрий Левин
Москва, ALT Linux Team
Проект: Sisyphus `http://git.altlinux.org`

Сборочная система git.alt: вчера, сегодня, завтра

История развития сборочной системы

- Доисторические приёмы формирования репозитория Sisyphus.
- Сборка репозиториев после внедрения `hasher` [Левин 2004].
- Сборочная система `git.alt` [Турбин 2008]: транзакционность и автономность.

Сборочная система с точки зрения пользователя

Жизненный цикл сборочного задания:
- Формирование.
- Обработка.
- Исправление нарушений.

Внутреннее устройство сборочной системы

- Структура сборочного задания.
- Приём задания на обработку.
- Сборка пакетов.
- Тестирование собранных пакетов.
- Формирование нового состояния репозитория.
- Тестирование нового состояния репозитория.
- Проверка ACL.
- Автоматическое закрытие ошибок.
- Обновление кеширующего репозитория.
- Публикация нового состояния репозитория.
- Обновление базы данных ACL.
- Рассылка уведомлений.

Перспективы дальнейшего развития

- Новые тесты отдельных пакетов и состояния репозитория.
- Поддержка персональных репозиториев.
- Дистрибутивность.

Литература

[Левин 2004] *Дмитрий Левин*, Hasher: технология безопасной сборки пакетов // Первая международная конференция разработчиков свободных программ на Протве. Тезисы докладов. М., 2004. С. 28–30.

[Турбин 2008] *Алексей Турбин*, Сборочная система git.alt // Пятая международная конференция разработчиков свободных программ на Протве. Тезисы докладов. М., 2008. С. 47–51.

Алексей Турбин
Рязань, ALT Linux Team
Проект: Sisyphus http://sisyphus.ru

Комплементарное хеширование подмножеств

Аннотация

Обсуждается разрешимость символов в программах, динамически компонуемых с разделяемыми библиотеками. Предложен новый вид зависимостей для rpm-пакетов. Рассмотрена возможность *комплементарного хеширования* — способа хеширования двух множеств R и P, который сохраняет возможность проверки $R \subset P$.

Зависимости на разделяемые библиотеки

Для запуска программы загрузчик `ld.so` конструирует исполняемый образ программы. При этом происходит загрузка необходимых разделяемых библиотек. Разделяемая библиотека характеризуется *именем* (soname). Кроме того, каждая разделяемая библиотека содержит экспортируемые *символы* (т. е. функции и глобальные переменные библиотеки). Для работоспособности программы существенной является не только возможность загрузки всех необходимых разделяемых библиотек по их имени, но и *разрешимость* всех символов, используемых в программе.

Зависимости rpm-пакетов напрямую отражают связи между программами и разделяемыми библиотеками, используя имена библиотек: пакет с разделяемой библиотекой предоставляет зависимость вида `Provides: libfoo.so.1`, а пакет, который использует библиотеку, требует зависимость `Requires: libfoo.so.1`. Однако информация о символах напрямую в зависимостях не используется.

Ясно, что зависимость на имя библиотеки может оказаться недостаточной — например, это можно быть связано с добавлением новых символов в библиотеку. Разработчик библиотеки, добавляя новые функции в библиотеку, считает, что он сохраняет обратную совместимость — существующие программы будут работать с новой версией библиотеки. Однако пользователь репозитория пакетов обычно хочет обновить интересную ему программу. Тогда в комбинации «новая

программа со старой библиотекой» могут образоваться неразрешимые символы (при обнаружении неразрешимых символов программа аварийно завершается).

Один из подходов, который позволяет в некоторой степени решить проблему обратной совместимости, — *версионирование символов* (набор новых символов библиотеки снабжается специальной меткой). Такой подход используется, в частности, в библиотеке `glibc` [DSO howto]. Однако этот подход нельзя автоматизировать, а из-за ограниченной совместимости он не получил широкого распространения.

Модель зависимостей с учетом символов

Символы ассоциируются с библиотеками, то есть считается установленным локальное соответствие между каждым экспортируемым символом и соответствующей ему библиотекой. (Формат ELF, вообще говоря, не требует такого соответствия. Однако подобная возможность реализована, например, в OpenSolaris [Direct Binding]).

Тогда пакет с разделяемой библиотекой предоставляет зависимость `Provides: libfoo.so.1` и множество экспортируемых символов этой библиотеки `foo1, foo2, foo3`. А пакет, который использует библиотеку, требует зависимость `Requires: libfoo.so.1` и множество используемых символов `foo1, foo2, ...`

Зависимость `Requires` считается удовлетворенной, если множество используемых символов является *подмножеством* экспортируемых символов (т. е. если все требуемые символы предоставлены).

Данная модель является слишком громоздкой. Но, строго говоря, хранить полный список символов нет необходимости, а нужно лишь уметь проверять, являются ли требуемые символы подмножеством предоставляемых. Возникает вопрос, нельзя ли придумать такую процедуру хеширования двух множеств R и P, при которой сохраняется возможность проверить вложение $R \subset P$?

Тогда для хранения информации о символах можно реализовать специальные *версии* rpm-зависимостей — т. н. set-версии, которые представляют собой захешированные множества символов: пакет с разделяемой библиотекой предоставляет зависимость вида `Provides: libfoo.so.1 = set:7f0252c3...`, а использующий библиотеку пакет требует `Requires: libfoo.so.1 >= set:3f5b289c...`

Комплементарное хеширование

Пусть заданы множества R и P, а также определен предикат $R \subset P$. *Схемой комплементарного хеширования* мы называем тройку $\langle H_R, H_P, \subset^* \rangle$, состоящую из двух функций хеширования $H_R(R) \to R^*$, $H_P(P) \to P^*$ и предиката $R^* \subset^* P^*$. При этом если $R \subset P$, то должно всегда выполняться и $R^* \subset^* P^*$. А если $R \not\subset P$, то $R^* \not\subset^* P^*$ выполняется с вероятностью $1 - \varepsilon$, где параметр ε задает одностороннюю ошибку, обусловленную потерей информации при хешировании. Односторонний характер ошибки означает, что проверка зависимостей никогда не будет давать ложных срабатываний, но может пропускать некоторые «настоящие» ошибки. Другими словами, в худшем случае проверка не сработает.

Оказывается, комплементарное хеширование в чистом виде невозможно: а именно, для $\varepsilon \ll \frac{1}{2}$ нельзя придумать хеш фиксированной длины. Будем требовать, чтобы для различных множеств R их хеш отличался (или «почти всегда» отличался). Тогда для представления такого хеша требуется хотя бы один бит на каждый всевозможный элемент (т. к. каждый элемент либо входит, либо не входит во всевозможные множества R). Другими словами, хеш не может быть короче битовой шкалы (которая, в свою очередь, не содержит избыточной информации). Кроме того, универсальной битовой шкалы не существует.

Таким образом, в название доклада вынесен негативный результат.

Компактное кодирование множества строк

Поскольку размер «хеша» растет пропорционально числу элементов множества, то можно кодировать каждый элемент отдельно, используя 16–32-битный хеш; а затем рассмотреть процедуру более эффективной упаковки элементов.

Приведем некоторые численные оценки. Пусть для представления множества P, состоящего из $1024 = 2^{10}$ символов, используется 20-битный хеш на элемент. 20-битный хеш позволяет адресовать 2^{20} элементов. Тогда вероятность коллизии с некоторым «случайным» символом, не входящим во множество P, составит $2^{10}/2^{20} = 2^{-10} \approx 0.1\%$.

Вычислим энтропию множества P. Множество P можно рассматривать как выбор 2^{10} элементов из 2^{20} элементов. Тогда энтропия

$log_2|P| = log_2\binom{2^{20}}{2^{10}} \approx 11710$ битов. Таким образом, оптимальный способ упаковки элементов может значительно снизить размера хеша (вместо 20 битов на элемент в упакованном виде потребуется примерно 11.5 битов на элемент).

В качестве процедуры упаковки элементов можно использовать, например, дельта-кодирование (предварительно выполнив сортировку элементов). Другая процедура кодирования, близкого к оптимальному, предложена Д. В. Чистиковым [Кодирование сочетаний]. Используется представление множества целых чисел в виде дерева, причем каждый узел дерева кодирует как значение элемента, так и (дополнительно) диапазоны значений «потомков».

Тестовая реализация

На момент публикации реализованы основные алгоритмы, необходимые для поддержки set-версий в зависимостях на разделяемые библиотеки, и выполнена тестовая пересборка репозитория. Нереализованной остается процедура упаковки элементов (set-версии представлены в упрощенном виде).

По результатам тестовой переборки удалось оценить «размерность» задачи. На архитектуре x86-64 репозиторий предоставляет 6740 разделяемых библиотек, причем каждая библиотека экспортирует в среднем 776 символов. Кроме того, 9947 пакетов в репозитории содержат зависимости на разделяемые библиотеки; каждый из них требует в среднем 9 библиотек для разрешения 269 символов. Таким образом, всего в репозитории нужно учитывать около 8 млн. символов. Тогда, если считать, что после упаковки каждый символ будет занимать примерно 12 битов, то для хранения информации о символах потребуется 12 Мб (на данный момент размер файла `pkglist.classic.bz2` — примерно 4 Мб).

Литература

[DSO howto] Ulrich Drepper. *How To Write Shared Libraries.* `http://people.redhat.com/drepper/dsohowto.pdf`

[Direct Binding] *Relocation Processing, Symbol Lookup, Direct Binding.* `http://docs.sun.com/app/docs/doc/817-3677/chapter3-2`

[Кодирование сочетаний] Д. В. Чистиков. *Кодирование сочетаний* (личное сообщение, ноябрь 2009 г.)

Игорь Власенко
Киев, ALT Linux

Проект: Автосборщики `http://www.altlinux.org/Gear/cronbuild`

Автоматизация сборки пакетов для дистрибутива

При сопровождении большого числа пакетов думать об автоматизации работы естественно. Рутинно повторяющаяся ручная работа порождает «эффект конвейера» — психический дискомфорт, ощущение малопродуктивного времяпровождения. Автоматизация этой работы позволяет существенно сэкономить время и получить удовольствие от творчества.

Есть и объективная потребность в автоматизации — среди дистрибутивов только единицы благодаря своему глобальному характеру могут позволить себе роскошь выбрасывать на ветер тысячи человеко-часов. ALT Linux – нишевый дистрибутив (русскоязычного сообщества), у него гораздо меньшее число потенциальных участников. Нехватка людей приводит к заброшенным / плохо сопровождаемым / отсутствующим в дистрибутиве пакетам. Однако, ALT Linux, похоже, нашел свой оригинальный путь — автоматизацию различных процессов сборки и тестирования, позволяющую даже неподготовленному человеку получить достаточно качественно собранные пакеты. Конечно, проблемы человеческого взаимодействия неизбежны, однако технологии все-таки берут на себя часть рабочей нагрузки. В русле таких тенденций развития дистрибутива возникает вопрос, нельзя ли автоматизировать сам процесс сборки.

Очевидно, что полностью автоматизировать процесс сборки невозможно, это задача на искусственный интеллект, разумная автоматизация — это автоматизация шаблонных действий, оставляющая нестандартные ситуации человеку.

Второе препятствие — часто обновления, хотя и имеют общую схему, но тяжело поддаются автоматизации. Пример: приложения, написанные на C/C++. Хотя сборка autotools-based приложений проходит по шаблону `configure && make && make install`, но как определить сборочное окружение и опции configure? Приходится смотреть в configure.ac, README и INSTALL. При этом человек может ошибить-

ся или пропустить часть работы, получив в итоге пакет с урезанной функциональностью, но, увы, механизировать это сложно.

Однако есть и пакеты, для которых механизировать сборку достаточно легко. В первую очередь это языки программирования с выраженной модульностью и централизованным репозиторием модулей, таким, как CPAN, CTAN, Cabal, ... Как правило, каждый модуль заворачивается в отдельный пакет, что в итоге дает сотни, а то и тысячи, мелких пакетов. Обновление этих пакетов достаточно шаблонно, естественно было бы поручить эту работу скрипту.

Восприятие автоматизации в коллективе

Поскольку массовая сборка по своей природе часто затрагивает чужие пакеты, с ней надо быть особенно осторожным. Автоматизация хороша, когда ее просят, либо в ней есть необходимость (заброшенные/недособранные пакеты) и нет возражений.

Обзор существующих «роботов»

- *jppimport* обновление java пакетов;
- *repocop-nmu* автоматизация QA;
- пока еще безымянный «робот» для обновления perl-пакетов;
- *girar-nmu* автоматизация массовых NMU и пересборок;
- *gear-cronbuild* автоматизация сборок по расписанию (снапшоты баз данных и др.).

Андрей Пономаренко, Владимир Рубанов
Москва, Институт системного программирования РАН

Проект: ABI Compliance Checker Project
http://ispras.linux-foundation.org/index.php/ABI_compliance_checker

Автоматизированный анализ обратной бинарной совместимости Linux библиотек

Аннотация

В работе рассматриваются вопросы обеспечения обратной совместимости при разработке новых версий библиотек. В качестве целевой среды рассматривается ОС Linux. Обратная совместимость между старыми и новыми версиями библиотек позволяет запускать приложения с использованием новых версий без пересборки приложения. В качестве решения проблемы в работе предложен подход на основе автоматизированного сравнения сигнатур функций и связанных с ними определений типов данных посредством анализа деревьев синтаксического разбора заголовочных файлов разных версий библиотек. В работе описаны принципы работы созданного в ИСП РАН инструмента для разработчиков библиотек, реализующего данный подход и результаты его апробации.

Системные библиотеки занимают важное место в архитектуре операционных систем и, в частности, в ОС Linux. Они являются основным фасадом, который используется приложениями для взаимодействия с системой, предоставляя приложениям набор программных интерфейсов (API). При этом у развивающихся системных библиотек новые версии появляются довольно часто — в среднем раз в несколько месяцев, и важной задачей при разработке таких обновлений является обеспечение *обратной совместимости* между различными версиями библиотек для минимизации негативного влияния на совместимость с существующими приложениями. Разделяется обратная совместимость на *уровне исходного кода* и на *бинарном уровне*. В первом случае для использования приложения с новой версией библиотеки его достаточно пересобрать (без изменения исходного кода), в случае бинарной совместимости приложение продолжает корректно работать с новой версией библиотеки без пересборки. Примером нарушения обратной бинарной совместимости может служить изменение размера

типа параметра функции или изменение структуры виртуальной таблицы класса в библиотеке. Такие изменения приводят к некорректной работе или разрушению приложений при работе с новыми версиями библиотек. В данной работе речь пойдет об обеспечении именно обратной бинарной совместимости.

Проблемы обратной совместимости возникают из-за изменений в коде библиотеки, которые могут быть преднамеренными и непреднамеренными. Разработчики библиотек стараются избегать непреднамеренных изменений, но, ввиду отсутствия подходящих автоматизированных подходов к анализу обратной совместимости, такие изменения все-таки происходят. Преднамеренные изменения в старых функциях, приводящие к потере обратной совместимости, допускаются разработчиками в случае крайней необходимости, если выгода за счет внесенных изменений важнее проблем из-за несовместимости с предыдущими версиями.

Известен ряд публичных историй, когда разработчики узнавали о возникновении проблем обратной совместимости уже после выпуска новой версии библиотеки. Например, в октябре 2007 года [1] разработчики стандартной библиотеки языка C++ libstdc++ случайно изменили порядок объявления методов в классе, что привело к изменению структуры виртуальной таблицы одного из классов и потере совместимости с предыдущими версиями библиотеки. В марте 2009 года [2] разработчики библиотеки FreeType2 преднамеренно изменили структуру одного из типов данных (структура PS_FontInfoRec_), что привело к слишком большому недовольству пользователей библиотеки из-за возникших проблем совместимости. В результате, в следующей версии разработчики библиотеки были вынуждены вернуть прежнюю структуру этого типа.

На основе анализа, проведенного авторами, было выделено 8 основных видов изменений в библиотеках, которые нарушают обратную бинарную совместимость. Анализ проводился на основе работ, посвященных правилам написания библиотек в Linux [3,4,5], и на основе изучения ABI наиболее распространенных архитектур центрального процессора, таких как x86 [6], x86-64 [7], PowerPC [8], Itanium [9,10], S390 [11] и ARM [12]. Результаты исследования представлены в таблице 1.

Предлагаемый подход к решению проблемы основан на сравнении сигнатур функций и определений типов данных посредством анализа деревьев синтаксического разбора соответствующих заголовочных

Таблица 1. Основные типы изменений, нарушающих обратную совместимость

Тип изменений		**Язык**	**Критичность**
Удаление функций или глобальных переменных		C/C++	Высокая
Изменение структуры виртуальной таблицы класса		C++	Высокая
Изменение числа или порядка параметров функции		C++	Высокая
		C	Средняя
Изменение типа данных параметра	Значительное	C/C++	Средняя
	Незначительное	C++	Средняя
	Незначительное	C	Низкая
Изменение типа данных возвращаемого значения	struct ↔ простой тип	C/C++	Средняя
	Значительное	C/C++	Средняя
	Незначительное	C/C++	Низкая
Удаление/добавление спецификатора “static”		C++	Средняя
Изменение значения enum, макроса или константы		C/C++	Низкая
Переопределение виртуальных функций		C++	Низкая

файлов двух версий библиотеки. Данный подход реализован инструментом для разработчиков библиотек, который носит название *abi-compliance-checker*.

Далее приводятся некоторые детали реализации этого инструмента. Для сравнения библиотек применяется утилита readelf. Данная утилита позволяет прочитать любой файл в формате ELF, в том числе и библиотеки. Из всей информации, которую можно получить при чтении бинарного кода библиотеки с помощью утилиты readelf, извлекается только список экспортируемых функций и их версии, если в библиотеке применяется версионирование функций. Затем два списка функций разных версий библиотек сравниваются и определяются добавленные и исключенные функции.

Для выявления проблем обратной совместимости, вызванных недопустимыми изменениями в типах данных или в сигнатурах функций, необходимо проанализировать изменения в заголовочных фай-

лах библиотеки, произошедшие в новой версии. Для сравнения двух заголовочных файлов сначала строятся соответствующие деревья синтаксического разбора кода этих файлов с помощью опции fdump-translation-unit компилятора gcc. Затем на основе деревьев разбора заголовочных файлов воссоздаются сигнатуры всех функций библиотеки с полным разбором строения типов параметров и возвращаемых значений.

Теперь, когда известна подробная информация о строении параметров функций библиотеки, производится анализ произошедших изменений в новой версии библиотеки.

На текущей стадии развития инструмент может находить все 8 видов проблем обратной совместимости, описанных ранее, и разбивать их на три группы с разными уровнями критичности (высокий, средний и низкий). Высокая критичность подразумевает гарантированное разрушение приложения при попытке вызвать соответствующую функцию, средняя критичность означает возможное разрушение данных, а низкая — признаки, которые означают потенциальное изменение семантики без явного изменения интерфейса.

Входными данными для инструмента являются два так называемых *дескриптора библиотеки*. Для каждой версии библиотеки нужно предоставить свой дескриптор. Дескриптор библиотеки представляет собой XML-файл следующего содержания:

```
<version>Версия</version>
<headers>Пути к заголовочным файлам</headers>
<libs>Пути к библиотекам</libs>
```

Секция <headers> включает в себя пути к директориям с заголовочными файлами библиотеки или к определенным заголовочным файлам. Секция <libs> содержит пути к директориям с библиотеками (файлы с расширением .so) или к конкретным библиотекам.

В сообществе разработчиков Linux под термином «библиотека» часто подразумевают не один отдельный файл с расширением .so, а набор таких файлов. Дескриптор библиотеки может описывать библиотеку в обоих случаях.

Пример дескриптора библиотеки Qt4.5.2, установленной в системную директорию /usr/local/qt4.5.2/, выглядит следующим образом:

```
<version>4.5.2</version>
<headers>/usr/local/qt4.5.2/include</headers>
<libs>/usr/local/qt4.5.2/lib</libs>
```

Таблица 2. Статистика обнаруженных проблем обратной совместимости

Тип изменений		**Критичность**	**Кол-во**	%
Удаление функций или глобальных переменных		Высокая	2617	27,0
Изменение структуры виртуальной таблицы класса		Высокая	64	0,7
Изменение числа или порядка параметров функции (C)		Средняя	163	1,7
Изменение типа данных параметра	Значительное	Средняя	1943	20,0
	Незначительное	Низкая	2136	22,0
Изменение типа данных возвращаемого значения	Значительное	Средняя	49	0,5
	Незначительное	Низкая	552	5,7
Удаление/добавление спецификатора “static”		Средняя	24	0,2
Переопределение виртуальных функций		Низкая	2151	22,2

Также дополнительными входными параметрами являются списки внутренних функций и типов, проверку которых осуществлять не требуется (например, функции, не предназначенные для использования в приложениях).

Выходными данными инструмента является отчет в формате html с результатами проверки двух версий библиотеки на совместимость. В отчете все типы проблем совместимости разделены на две группы — проблемы функций и проблемы типов данных. При этом за каждой проблемой, связанной с изменением типа данных, закреплен список функций, на которые это изменение повлияло.

Инструмент abi-compliance-checker был опробован на 60 основных библиотеках Linux (45 C и 15 C++). В совокупности было проверено 1460 версий библиотек и найдено 9699 изменений в их бинарном интерфейсе (кумулятивно от версии к версии). Анализ преднамеренности внесения данных изменений не проводился. Более детальная статистика по различным типам обнаруженных изменений приведена в таблице 2.

Инструмент принят в репозитории Debian, FreeBSD, Gentoo, Haiku, Maemo, Mandriva и Ubuntu, а также добавлен в качестве дополнительного шага проверки в процедуре инсталляции, используемой apt и rpm5. Разработчики библиотеки libssh утвердили abi-compliance-checker в качестве обязательного средства контроля качества в основном цикле разработки.

Литература

[1] Libstdc++ ABI break, http://gcc.gnu.org/ml/libstdc/2007-10/msg00041.html.

[2] ABI breakage in Freetype2 2.3.8, http://www.mail-archive.com/freetype-devel@nongnu.org/msg03192.html.

[3] Mike Hearn, “Writing shared libraries”, http://plan99.net/~mike/writing-shared-libraries.html.

[4] KDE TechBase, “Policies/Binary Compatibility Issues With C++”, http://developer.kde.org/documentation/other/binarycompatibility.html.

[5] Linux.org, “Program Library HOWTO”, http://www.linux.org/docs/ldp/howto/Program-Library-HOWTO/shared-libraries.html.

[6] The System V Application Binary Interface Intel386 Architecture Processor Supplement, http://refspecs.freestandards.org/elf/abi386-4.pdf.

[7] The System V Application Binary Interface AMD64 Architecture Processor Supplement, http://www.x86-64.org/documentation/.

[8] The System V Application Binary Interface PowerPC Processor Supplement, http://refspecs.freestandards.org/elf/elfspec_ppc.pdf.

[9] Itanium C++ ABI, http://www.codesourcery.com/public/cxx-abi/abi.html.

[10] Intel Itanium Processor specific ABI, http://refspecs.freestandards.org/elf/IA64-SysV-psABI.pdf.

[11] LINUX for S/390. ELF Application Binary Interface Supplement, http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l390abi0.pdf.

[12] Application Binary Interface for the ARM Architecture, http://infocenter.arm.com/help/topic/com.arm.doc.ihi0036b/IHI0036B_bsabi.pdf.

Андрей Пономаренко, Владимир Рубанов,
Алексей Хорошилов
Москва, Институт системного программирования РАН

Проект: API Sanity Autotest
http://ispras.linuxfoundation.org/index.php/API_Sanity_Autotest

Автоматическая генерация тестов для C/C++ библиотек

Аннотация

В докладе представлен инструмент для автоматической генерации тестов работоспособности на основе информации, автоматически извлекаемой из заголовочных файлов C/C++ библиотек. Ключевой особенностью инструмента является возможность улучшения качества генерируемых тестов за счет аннотации типов и параметров функций дополнительной семантической информацией. Эта возможность позволяет значительно сэкономить усилия при разработке тестов для тестирования больших библиотек, в которых типы данных и другие семантические элементы являются общими сразу для многих функций. В дополнение к тестированию работоспособности сгенерированные тесты могут быть использованы в качестве хорошей стартовой точки для разработки полноценных функциональных тестов.

Не секрет, что разработка тестов — не самое популярное занятие в мире свободного программного обеспечения. Поэтому неудивительно, что многие системные библиотеки не имеют тестов, которые могли бы обеспечить даже минимальную проверку корректности работы библиотеки в результате внесения в нее изменений или портирования в новое окружение. Так из 711 src.rpm пакетов, имя которых начинается с префикса lib, обнаруженных в репозитории Sisyphus 7 июля 2010 года, порядка 260 не содержит вообще никаких тестов. Более того, большинство тестов, которые присутствуют в пакетах, содержат лишь регрессионный набор для проверки отсутствия ранее обнаруженных ошибок. А систематические тесты, которые покрывали хотя бы наиболее важные части библиотеки, доступны лишь для небольшого числа наиболее зрелых проектов.

Большинство СПО проектов просто не имеют необходимых ресурсов, чтобы уделять разработке тестов достаточное внимание, особенно на начальных стадиях. Да и по мере роста ситуация не становится проще, так как задача покрытия тестами всех функций библиотеки становится всё менее обозримой.

В данных условиях значительную помощь в исправлении ситуации может оказать инструмент API Sanity Autotest, разрабатываемый в Институте системного программирования РАН. Этот инструмент способен полностью автоматически генерировать тесты работоспособности[1] только на основе заголовочных файлов библиотеки. При этом используется информация о сигнатуре публичных функций библиотеки, то есть о типах её входных и выходных параметров. Конечно, в большинстве случаев этой информации недостаточно для создания корректных тестов, например, из-за необходимости инициализации библиотеки или значений определенного типа неким нетривиальным образом.

Поэтому API Sanity Autotest поддерживает возможность задания дополнительной семантической информации об особенностях библиотеки, представленных в ней типах данных и специфики их использования в определенных функциях. Типичными примерами такой информации являются:

- описание, как получить корректное значение определенного типа данных;
- описание, каким должно быть корректное значение определенного параметра функции;
- описание, какие проверки можно сделать для возвращаемых значений определенного типа.

Казалось бы, мы варим суп из топора: какая автоматизация, если требуется вручную описать код, необходимый и при обычной разработке тестов. Но тем не менее, польза от API Sanity Autotest есть как минимум по двум направлениям.

Во-первых, описания пишутся в виде кода на C/C++, в котором могут присутствовать специальные конструкции, при помощи которых можно попросить инструмент сгенерировать код для получения

[1] Под *тестами работоспособности* обычно понимают тесты, проверяющие только, что основные функции системы выполняются более-менее правильно, то есть, что система не разрушается и возвращает результаты, проходящие простейшие проверки на корректность (полная проверка при этом не выполняется).

значения определенного типа или подготовить параметры и вызвать определенную функцию.

Во-вторых, семантическая информация одновременно привязывается ко многим местам, где она требуется. Например, информация о специфике инициализации библиотеки записывается один раз для всех функций. Информация о создании объектов определенного типа также записывается в одном месте и затем используется для всех функций, у которых есть параметр такого типа, и в других необходимых местах, таких как специальные конструкции, о которых шла речь выше.

Всё вместе это позволяет минимизировать дублирование кода, необходимого для подготовки описаний, что значительно сэкономит усилия при генерации тестов для больших библиотек, в которых типы данных и другие семантические элементы являются общими сразу для многих функций.

Полученные в результате генерации тесты содержат минимальные проверки того, что функция работает на наиболее простом сценарии ее использования и возвращает более-менее правильный результат. Наличие таких тестов уже является большим шагом вперед по сравнению с отсутствием тестов вовсе. И даже такие тесты позволяют обнаружить некоторое число ошибок.

Но не менее ценным фактом является то, что сгенерированные тесты могут быть использованы в качестве хорошей стартовой точки для разработки полноценных функциональных тестов. Когда тесты работоспособности доведены до состояния корректной работы, то есть когда семантической информации достаточно для генерации корректных вызовов всех необходимых функций, полученный тестовый набор можно доработать вручную для обеспечения более качественного тестирования наиболее важных частей библиотеки. Для этих целей API Sanity Autotest поддерживает генерацию тестов в формате фреймворка Template2Code (`http://sourceforge.net/projects/template2code/`), который предоставляет удобные возможности для дальнейшего развития тестового набора.

API Sanity Autotest реализован в виде Perl-скрипта, который извлекает информацию о публичных функциях из списка экспортируемых функций разделяемой библиотеки с помощью readelf. Информация о сигнатурах функций извлекается из заголовочных файлов при помощи компилятора gcc. Таким образом, список зависимостей инструмента весьма небольшой: gcc, binutils и perl.

Инструмент уже используется в ряде открытых и проприетарных проектов. В частности, он интегрирован в make check для rpm5.org (`http://rpm5.org/cvs/dir?d=rpm/build/auto`).

Существует и много идей по развитию инструмента. В частности, в настоящее время ведется Google Summer of Code проект по автоматическому извлечению семантической информации из splint-аннотаций кода заголовочных файлов, если таковые там найдутся. Другим направлением для развития является генерация множества тестовых испытаний для одной функции за счет использования нескольких наборов параметров, сформированных по различным принципам.

Михаил Пожидаев, Анатолий Камынин
Томск, Самара, ALT Linux Team

Проект: ALT Linux Homeros `http://homeros.altlinux.org`, `http://www.tiflocomp.ru`

Развитие окружения ALT Linux Homeros для лиц с ограничениями по зрению

Аннотация

Доклад освещает ход работы по созданию специализированного дистрибутива на базе систем ALT Linux для лиц с ограничениями по зрению. Описываются достижения, проблемы и планы на будущее. Приводится краткая информация об аналогичной иностранной разработке Vinux.

Продолжает активно развиваться проект ALT Linux Homeros, целью которого является разработка дистрибутивов для людей с нарушениями зрения.

Самый ответственный компонент в системах с речевым выводом — речевой сервер, обеспечивающий взаимодействие программ экранного доступа и синтезаторов речи. Сервер должен иметь гибкую систему конфигурирования и, чтобы упростить процедуру настройки, охватывать максимальное число параметров вывода речи.

Больше года работы ушло на создание новой переработанной версии сервера VoiceMan-1.5.0. Текущая версия разрабатывалась с учётом отзывов, полученных при публикации первых пробных говорящих дистрибутивов Homeros в конце 2008 г.

VoiceMan выполняет автоматическое переключение синтезаторов для обработки различных языков (английского, русского, в будущем также украинского) и для голосового выделения разных элементов форматирования.

Новый VoiceMan обладает способностью горячей перезагрузки параметров без потери клиентских соединений, что очень востребовано при работе без зрительного контроля.

Помимо речевого сервера в специализированном дистрибутиве играет важную роль аудио-проигрыватель. Кроме своих основных функций, он должен решать задачу удобного воспроизведения аудиокниг. В настоящий момент обсуждаются различные варианты его реализации.

Актуальной остаётся тема русификации речевого синтезатора espeak. Espeak — это полностью свободный синтезатор речи, поддерживающий много языков, среди которых есть и частичная поддержка русского, реализованная приблизительно на 50%.

Текущая реализация русификации espeak создана группой разработчиков из Томского государственного университета, в которую входит преподаватель фонетики с филологического факультета Дмитрий Катунин и выпускница факультета информатики Рената Пожидаева. Для завершения этой работы проект Homeros приглашает всех заинтересованных людей.

В настоящее время к выпуску готовятся два дистрибутива: ALT Linux Homeros Desktop на основе оконной системы GNOME и ALT Linux Homeros Friend на основе emacspeak.

ALT Linux Homeros Desktop должен содержать традиционный набор настольных озвучиваемых приложений: OpenOffice.org, Firefox, Thunderbird и др. Он является типовым решением, привычным для большинства пользователей.

Среда emacspeak, которая должна быть основой ALT Linux Homeros Friend, менее привычна для работы и требует дополнительного времени для знакомства.

Главными её достоинствами являются более скромные требования к аппаратному обеспечению и более высокая производительность. Emacspeak хорошо подходит для работы с профессиональными издательскими системами LaTeXи lilypond (издательская система для вёрстки музыкальных партитур), которые доступны незрячим пользователям — входные материалы являются просто текстовыми файлами.

Главные недостатки emacspeak заключаются в неудовлетворительной поддержке редактируемых офисных документов и текстовом веббраузере, не имеющем поддержки Java Script. Предполагается компенсировать эти недостатки за счёт использования API для «облачных» служб, которые предоставляют такие проекты как Google Cl и g-client.

В качестве инструмента для установки систем на ПК сейчас рассматривается возможность реализации комплекта скриптов, переносящих среду с LiveCD.

Проект ALT Linux Homeros сотрудничает с единственной аналогичной развивающейся иностранной разработкой — Vinux. Vinux версии 3 — это измененный вариант популярного дистрибутива Ubuntu 10.04 Lucid Lynx, подготовленный с учётом потребностей лиц с нарушениями зрения.

Разработчики Vinux стремились обеспечить пользователя доступной средой с речевой поддержкой уже на этапе загрузки системы; оптимизировать внешний вид рабочего стола GNOME; пополнить дистрибутив и репозитарий пакетами, более подходящими для работы с программами чтения экрана. Надо сказать, что многое им удалось.

Vinux включает три программы экранного доступа (Orca для среды GNOME; Speakup и Yasr для текстовой консоли), две полноэкранные лупы (одна входит в состав Orca, другую предоставляет Compiz Window Manager). Поддерживаются брайлевские дисплеи (пакет Brltty). По умолчанию используется синтезатор речи Espeak и речевой сервер Speech-Dispatcher.

При загрузке Vinux сразу включается речевое сопровождение, а после загрузки рабочей среды GNOME автоматически запускается Orca. Таким образом, пользователь попадает в среду с речевой поддержкой и доступом к элементам пользовательского интерфейса. Это реализовано за счёт использования библиотеки GTK+, доступной для вспомогательных технологий. Кроме того, есть возможность «превратить» уже установленный дистрибутив Ubuntu в Vinux, установив соответствующие пакеты из репозитория Vinux.

В ходе сотрудничества проектов Homeros и Vinux предполагается совместная подготовка специализированных пакетов, а также обмен идеями и решениями в области вспомогательных технологий Linux.

Дмитрий Сподарец
Украина/Одесса, ОНУ имени И. И. Мечникова
Проект: Аппаратно-программный комплекс TERM

Расширение возможностей аппаратно-программного комплекса TERM для исследования низкотемпературной плазмы

Аннотация

В работе описываются новые возможности аппаратно-программного комплекса TERM для проведения исследований низкотемпературной плазмы спектральным методом.

Разработка перспективных технологических процессов требует исследования самоорганизации гетерогенных плазменных структур, таких как пылевая или плазма продуктов сгорания. Для проведения подобных исследований необходимо создавать новые средства диагностики с возможностью регистрации кратковременных процессов, а также обработки данных в реальном времени.

Разработанный ранее аппаратно-программный комплекс TERM-1 [1] позволяет проводить исследования температур пламени методом термопары и относительной яркостной пирометрии (рис. 1). Так как оптические методы диагностики пламени и плазмы базируются на схожей теории, была поставлена задача расширить возможности аппаратно-программного комплекса TERM для проведения оперативной диагностики таких параметров низкотемпературной плазмы, как температура газовой и конденсированной фаз, концентрации электронов.

Для определения температуры конденсированной фазы можно использовать сплошной спектр излучения, который описывается формулой Вина:

$$I_{\lambda T} = \varepsilon_{\lambda T} \frac{C_1}{\lambda^5} \exp\left(-\frac{C_2}{\lambda T_{\text{я}}}\right) \tag{1}$$

Предполагая, что $\varepsilon_{\lambda T} = const$, выражение (1) можно привести к виду:

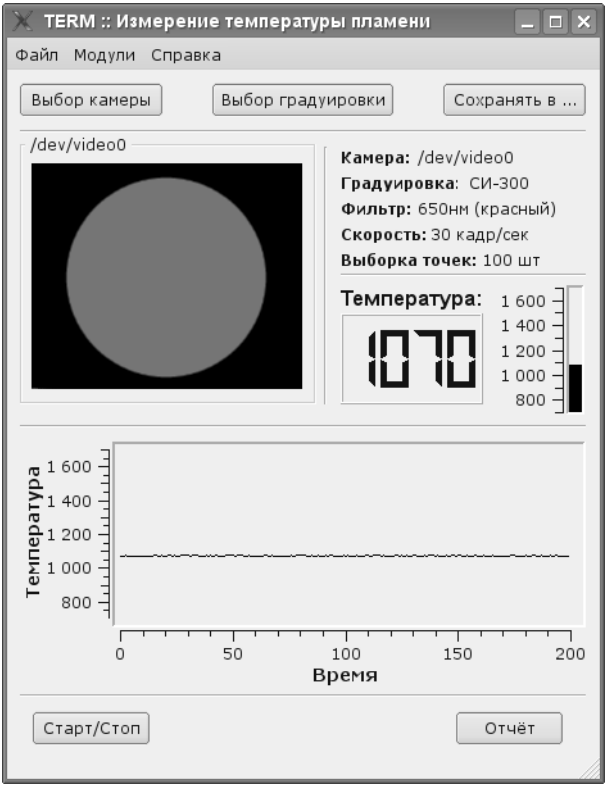

Рис. 1. Рабочее окно TERM для определения температуры пламени

$$F_{\lambda T} = \lg I_{\lambda T} + 5 \lg \lambda = const - \frac{\lg e \cdot C_2}{\lambda T} \tag{2}$$

откуда следует, что зависимость $F_{\lambda T}$ от $\frac{1}{\lambda}$ даёт прямую линию, тангенс угла которой позволяет определить температуру по формуле:

$$T = \frac{6241}{\operatorname{tg} \alpha} \tag{3}$$

Для измерения температуры газовой фазы продуктов сгорания применяется метод относительной интенсивности спектральных линий, которые можно выразить в виде:

$$I=\frac{hc}{\lambda}A_{ik}\frac{g_i}{g_0}N_0 e^{\frac{E_i}{KT}} \tag{4}$$

Если взять отношение интенсивности двух спектральных линий, то температура определяется по формуле:

$$T=\frac{E_2-E_1}{k\ln\left(\frac{I_1}{I_2}\frac{\lambda_1 A_2 g_2}{\lambda_2 A_1 g_1}\right)} \tag{5}$$

Концентрацию электронов определяем используя формулу Штарковской полуширины спектральной линии [2]:

$$\Delta\lambda_{\text{шт}}=2\left(1+1,75\cdot 10^{-4}Ne^{1/4}\alpha\left(1-0,068Ne^{1/6}T^{-1/2}\right)\right)10^{-16}\omega Ne \tag{6}$$

Таким образом, используя сплошной и линейчатый спектры, полученные при помощи дифракционного спектрографа, и регистрируя его вебкамерой с передачей сигнала на ЭВМ, производим регистрацию параметров плазмы в реальном времени. Для обработки видеоряда, приходящего с вебкамеры, был разработан дополнительный программный модуль для TERM (рис. 2).

Обработка видеопотока, приходящего с вебкамеры, выполняется при помощи библиотеки компьютерного зрения с открытым исходным кодом — OpenCV. Эта библиотека позволила воспользоваться последними наработками робототехники в области анализа видеопотока.

Использование свободного программного обеспечения в данном комплексе позволяет оперативно адаптировать его под различного рода задачи и экономить средства на приобретении аналогичного дорогостоящего ПО.

Возможна замена вебкамеры на скоростную видеокамеру, которая позволит в реальном времени регистрировать кратковременные процессы.

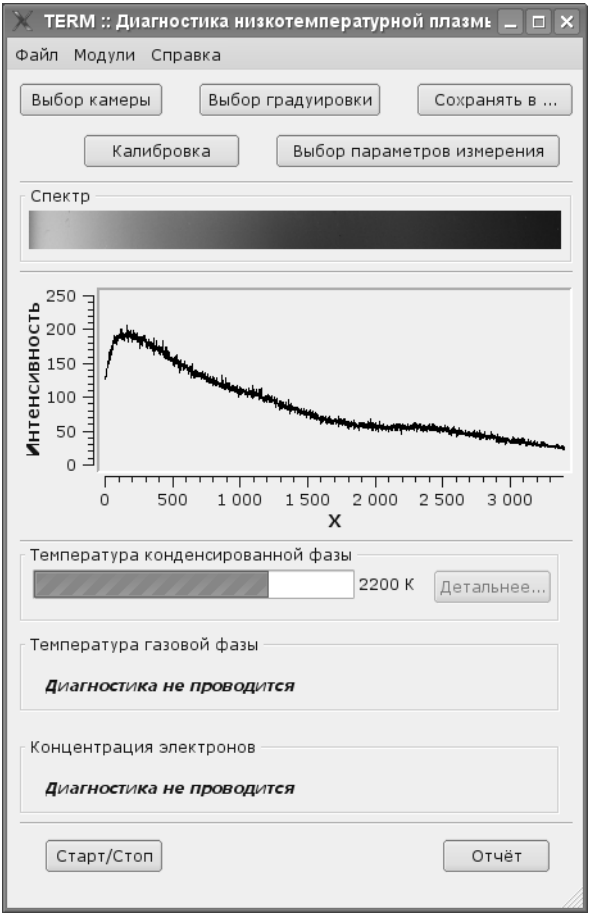

Рис. 2. Рабочее окно TERM для проведения диагностики низкотемпературной плазмы

Литература

[1] Сподарец Д. В. Разработка аппаратно-программного комплекса TERM-1 по измерению температур быстропротекающих процессов на основе открытого программного обеспечения // Сборник тезисов 5-й международной конференции разработчиков свободных программ, г. Обнинск. М.: 2008. С. 78-79.

[2] Драган Г.С. Електрофізичні властивості двофазної плазми продуктів згорання. Дис. на здоб. вчен. ступеня канд. фіз.-мат. наук, Одеса 1980.

Евгений Чичкарёв
Мариуполь, Украина, Приазовский государственный технический университет

Разработка сервера вычислений и web-интерфейса для удаленной работы с системами компьютерной математики в среде пакета Moodle

Аннотация

Данная работа посвящена разработке веб-интерфейса для удалённого доступа к вычислительному ядру систем компьютерной математики (Maxima, SciLab) из СДО Moodle. Представлены результаты исследования времени отклика при решении математических и вычислительных задач различной сложности, проанализированы различные варианты организации запросов к вычислительному ядру СКМ, а также создания виртуальных лабораторных работ в среде Moodle.

Современное дистанционное обучение строится на использовании следующих основных элементов:

- среды передачи информации (почта, телевидение, радио, информационные коммуникационные сети);
- методов, зависимых от технической среды обмена информацией.

Использование технологий дистанционного обучения позволяет:

- снизить затраты на проведение обучения (не требуется затрат на аренду помещений, поездок к месту учебы, как учащихся, так и преподавателей);
- проводить обучение большого количества человек;
- повысить качество обучения за счет применения современных средств, объемных электронных библиотек и т. д.;

– создать единую образовательную среду (особенно актуально для корпоративного обучения).

Использование дистанционного обучения для преподавания различных разделов математики и учебных дисциплин, включающих элементы математического моделирования, позволяет усилить их прикладную и практическую направленность и создает условия для реализации индивидуального подхода на качественно новом уровне.

На данный момент существует множество веб-интерфейсов к различным информационным системам, в т. ч. и математическим пакетам. Большинство коммерческих пакетов, таких как MatLab, Maple, MathCad, Mathematica, обладают интерфейсами, позволяющими использовать их функциональность удаленно. Другой характерный пример — Mathcad Calculation Server, позволяющий работать с MathCad-документами в сети. Аналогичные расширения существуют и для Maple (MapleNet) [1] или Mathematica (webMathematica).

С появлением системы дистанционного образования Moodle все больше проектов организации ДО стали ориентироваться на данную систему как на функциональную оболочку. Принципы функционирования системы Moodle позволили с достаточной легкостью вести разработку, ориентируясь изначально на использование в качестве базовых функций API данной системы как базовой среды для интеграции. В частности, появились проекты, которые являются надстройками над системой дистанционного обучения Moodle.

Web-интерфейс к среде MatLab позволил разработчикам создать проект, который являлся модулем фильтра в Moodle и использовал MWS как шлюз к вычислительным способностям MatLab. Данный фильтр позволял встраивать команды MatLab в документы и выполнять их перед передачей на сторону клиента (в браузер пользователя) [2, 3]. Однако следует отметить, что компания MathWorks отказалась от своего проекта MWS, в результате чего он стал недоступен для последних версий MatLab.

Существуют проекты, основанные на использовании СДО Moodle и MapleNET (см. проект AiM, `http://www.colum.edu`). Данное решение было разработано уже на известной идее «встраивания» команд в документы курсов, лабораторных работ и тестов, и вычисления их с последующим отображением результата.

Технологически средства организации удалённого доступа несколько различаются, но характерным примером может быть webMathematica, основанная на технологиях сервлетов и JSP.

В рассматриваемой области существуют решения, базирующееся исключительно на opensource-программных продуктах, а именно — СДО Moodle и математический пакет Maxima (пример — пакет STACK, System for Teaching and Assessment using a Computer algebra Kernel, см. `http://stack.asu.edu/`). Для вычисления выражений и построения графиков авторы STACK использовали вычислительное ядро СКМ Maxima. Для отображения результатов, если они представлены в текстовом виде, используется библиотека MathML и программа Gnuplot для работы с изображениями и преобразованием их в файлы формата «png». Сама среда STACK написана с использованием скриптового языка PHP и использует СУБД MySQL, благодаря чему данная система успешно работает с платформами Linux, Windows, Mac OS X.

Возможности удалённой работы с вычислительными пакетами обеспечивает и пакет Sage (http://sagemath.org/). При помощи Sagenotebook можно удалённо работать с целым рядом математических пакетов, входящих в состав Sage, а также публиковать в Интернет интерактивные документы.

Учитывая технологию реализации СДО Moodle (пакет разработан на PHP), при запуске вычислительного ядра того или иного пакета на сервере важной проблемой является фильтрация команд пользователя для блокирования потенциально опасных функций или команд конкретного пакета, требующих доступа к файловой системе сервера. Кроме того, в ряде открытых реализаций веб-интерфейса к математическим пакетам для обращения к ядру пакета используются потенциально опасные функции php — exec(), passthru() и т. п. Наконец, серьёзной проблемой может являться масштабирование проекта с ростом объёма использования СДО.

В данной работе проведено исследование особенностей работы удаленного доступа к СКМ Maxima из документов, размещённых в СДО Moodle.

Для проведения исследований была создана среда, представляющая собой сегмент сети. Для регулирования параметров сервера была использована его виртуализация средствами программного обеспечения VMware Workstation. На виртуальном сервере были установлены веб-сервер Apache, СУБД MySQL, СДО Moodle. Виртуальная среда соединялась с операционной системой-хостером путем эмуляции сети с коэффициентом потери пакетов, равным 0,001%. Операционная система-хостер так же выполняла роль моста для соединения (уже

Таблица 1. Время вычисления и объем оперативной памяти для задач различной сложности

Сложность задачи	Объем занимаемой памяти (Мб)	Время отклика (сек)
сложение	2,014	0,555
интеграл	29,673	1,754
сумма ряда	2,517	0,727
система ОДУ (2)	46,862	3,709
система ОДУ (4)	50,231	4,104

через физическую среду) с другими рабочими станциями (на момент проведения эксперимента их было четыре).

Компоненты системы (apache, ядро Moodle, ядро MySQL) без нагрузки на них занимают в оперативной памяти сервера менее 90 Мб.

Для исследования параметров нагрузки сервера при различных типах задач, выполняемых пакетом MAXIMA, было выделено 5 задач разной степени сложности:

- сложение двух чисел;
- вычисление нетабличного интеграла;
- вычисление суммы ряда;
- расчет решения системы двух обыкновенных дифференциальных уравнений;
- расчет решения системы четырех обыкновенных дифференциальных уравнений.

Результаты экспериментов приведены в таблице.

Для исследования работы сервера с различной нагрузкой были созданы два варианта виртуального сервера с выделением оперативной памяти 512 Мб и 1024 Мб ОЗУ соответственно (тестирование проводилось на ноутбуке с одноядерным процессором и тактовой частотой 1,86 GHz).

Обращение к вычислительному ядру Maxima осуществлялось из удаленной среды с помощью серверного скрипта на php. В результате проведения вычислительного эксперимента установлено, что вне зависимости от количества параллельных процессов Maxima объем оперативной памяти, занимаемый каждым из них, изменяется незначительно. Незначительные изменения обусловлены влиянием на систему фоновых задач операционной системы, в рамках которой проводился

эксперимент. Интервал между запусками экземпляров не превышал 0,023 секунды, что существенно меньше времени вычисления тестовой задачи (около 1,5 с для единичного экземпляра Maxima).

Установлено, что время отклика СКМ линейно растёт по мере роста количества одновременно запущенных экземпляров с резким увеличением при предельных значениях величин ресурсов, выделяемых системе на обработку задач.

Для сервера с двухядерной архитектурой (1024 Мб ОЗУ, тактовая частота 2 GHz) был отмечен куда менее резкий рост времени отклика по мере увеличения числа параллельно запущенных процессов.

Для запуска Maxima на реальном сервере СДО Moodle использовалось несколько вариантов:

– с использованием функции php passthru();

– с использованием сокетов.

Последний вариант является предпочтительным, т. к. позволяет более удобно контролировать нагрузку сервера без резкого роста времени отклика всех запущенных процессов.

Для организации виртуальных и практических работ в СДО Moodle элементы графического интерфейса пользователя, реализованные с использованием технологии Ajax. В систему Moodle включены изначально несколько библиотек, необходимых для создания интерфейса и ряда модулей:

– официальная библиотека JavaScript для Moodle — YUI (разработка Yahoo Developer Network; представляет собой набор утилит и элементов управления, написанных с использованием JavaScript и CSS, для создания интерфейса веб-приложений с использованием таких методов, как DOM-сценарий, DHTML и AJAX);

– собственная библиотека JavaScript-кода (lib/JavaScript-static.js);

– редактор TinyMCE и его библиотеки для работы с HTML-документами внутри среды Moodle используется компоненты библиотеки.

Аналогичный подход к организации веб-интерфейса и выполнения вычислений в документах Moodle может использоваться и для других СКМ (Octave, Scilab и т. п.).

Однако, как показали результаты исследования работы виртуального сервера и реального сервера с работающей СДО Moodle, к удаленному запуску вычислительных приложений следует относиться с

осторожностью, т. к. успешное выполнение расчётов требует использования мощного сервера, оснащённого достаточным объемом памяти и производительным многоядерным процессором.

Использование удаленного доступа к СКМ для подготовки иллюстраций, проведения виртуальных лабораторных и практических работ и т. п. с предопределённым набором запросов позволяет существенно упростить требования к безопасности и фильтрации команд СКМ.

В настоящее время разработывается комплект документации для организации учебного процесса на базе СКМ (методические указания по различным дисциплинам, руководство по лабораторному практикуму и т. п.).

Дистанционное применение СКМ в преподавании математических курсов способствует интересу студентов к изучаемому с их помощью материалу. Высочайшая степень визуализации вычислений, позволяет легче понять математическую (или физическую — при изучении вопросов моделирования) сущность изучаемых методов и алгоритмов. Многим учащимся интересно изучение самих этих систем, которые являются крупными интеллектуальными программными продуктами. Кроме того, использование СКМ позволяет сделать процесс обучения более быстрым и насыщенным, лучше разместить в рамках учебного графика предусмотренный программой объем материалов, обогащать читаемые курсы новыми методами и понятиями.

Литература

[1] Дьяконов В.П. Maple 10 в математике, физике и образовании / В. П. Дьяконов — М: Солон-Пресс, 2006. — 720 с.

[2] Котельников И.А. MATLAB Web Server: вычисления в Интернете / Котельников И.А., Черкасский В.С. // Математика в приложениях. 2005. — № 1 (5) / 2004. — С. 4—11.

[3] Котельников И.А. Разработка и использование моделирующих программ для дистанционного и самостоятельного обучения с использованием Matlab Web Server / Котельников И.А., Матвеенко А.Н., Черкасский В.С. // Тезисы докл. международной науч.-метод. конф. «Новые информационные технологии в университетском образовании» (Новосибирск, 23—24 сентября 2003). — Новосибирск: СибГУТИ, 2003. С. 92—94.