УДК 65.011.56

В. Д. Бойко, канд. техн. наук

ПРАКТИЧЕСКИЕ АСПЕКТЫ ИЗУЧЕНИЯ ИНФРАСТРУКТУРЫ КРУПНЫХ IT-ПРОЕКТОВ

Аннотация: *Рассмотрены практические аспекты организации взаимодействия в крупных IT-проектах. Выделены основные составляющие инфраструктуры, обеспечивающей создание IT-продуктов. Приведен обзор существующих программных решений, а также рекомендации по изучению инфраструктуры IT-проектов в вузах. Основное внимание уделено стеку программного обеспечения с открытым исходным кодом.*

Ключевые слова: *информационные технологии, IT-проект, IT-продукт, IT-инфраструктура, организация проекта, система контроля версий, багтрекер, система автодокументирования, исходные коды, веб-платформа, практики IT-разработки, конструктивизм, обучение программированию на примерах, программная инженерия, информационная инженерия, открытое программное обеспечение*

V. Boyko, PhD.

TEACHING AND LEARNING PRACTICAL ASPECTS OF LARGE IT-PROJECTS INFRASTRUCTURE

Abstract. *The paper presents an analysis of practical aspects of interaction in large IT-projects. These have included the basic components of the infrastructure for the creation of IT-products. Then the paper reviews existing software solutions, as well as recommendations for the study of IT-infrastructure projects in universities with emphasis on an open-source stack of software.*

Keywords: *information technology, IT-project, IT-products, IT-infrastructure, project organization, version control system, bug tracker, self-documentation system, source code, web-based platform, practice of IT-development, constructivism, example-based programming, software engineering, open source software, information engineering.*

В. Д. Бойко, канд. техн. наук

ПРАКТИЧНІ АСПЕКТИ ВИВЧЕННЯ ІНФРАСТРУКТУРИ ВЕЛИКИХ IT-ПРОЕКТІВ

Анотація. *Розглянуто практичні аспекти організації взаємодії у великих IT-проектах. Виділено основні складові інфраструктури, що забезпечує створення IT-продуктів. Приведений огляд існуючих програмних рішень, а також рекомендації по вивченню інфраструктури IT-проектів у ВНЗ. Основну увагу приділено стеку програмного забезпечення з відкритим джерельним кодом.*

Ключові слова: *інформаційні технології, IT-проект, IT-продукт, IT-інфраструктура, організація проекту, система контролю версій, багтрекер, система автодокументування, джерельні коди, веб-платформа, практики IT-розробки, конструктивізм, вивчення програмування за прикладами, програмна інженерія, інформаційна інженерія, відкрите програмне забезпечення*

Актуальность работы. Индустрия информационных технологий (IT-индустрия) является наиболее высокотехнологичной и высокорентабельной сферой экономики Украины. Ее отличие от прочих секторов индустрии состоит в том, что она не требует больших капиталовложений со стороны государства, является экологически чистой и при этом способствует развитию научного, интеллектуального и технологического потенциала страны.

По состоянию на 2012 год на рынке IT-индустрии функционировало две тысячи компаний, в которых было занято более 150 тысяч человек. По некоторым экспертным оценкам, валовый доход компаний IT-индустрии составлял более 15 млрд. грн. с ростом в 30 – 40 % ежегодно. При этом около 80 % общего дохода отрасли приходилось на экспортные услуги [1]. Отрасль демонстрирует рост даже в условиях кризиса (по разным оценкам, около 8 % за 2014 год). Такое положение вещей предъявляет повышенные требования к эффективности образования (подготовки) IT-специалистов.

На первом этапе развития IT-индустрии в создании информационного продукта (IT-продукта) участвовал коллектив из нескольких человек.

Глобализация IT-процессов, увеличение масштаба проектов и аутсорсинг привели к тому, что в разработке отдельного IT-продукта, задействованы коллективы из сотен (в отдельных случаях – тысяч) специалистов различной специализации и направлен-

ности (разработчики, тестировщики, эксперты по юзабилити, авторы документации, администраторы и так далее), которые распределены между городами и странами и вынуждены взаимодействовать в разных временных зонах. Это обусловливает важность организации и построения взаимодействия между разработчиками крупных проектов, поскольку невнимание к этой проблеме может сказаться на качестве продукта и сроках его создания [2; 3].

Эффективное взаимодействие при создании IT-продуктов предполагает понимание участниками проекта основных концепций взаимодействия, а также владение специфическими знаниями, навыками и инструментами для работы в большом распределенном коллективе. Освоение перечисленных концепций и инструментария позволит выпускнику учебного заведения эффективно выполнять свои задачи в коллективе разработчиков IT-продукта и повысит его конкурентоспособность на рынке труда.

Целью статьи является обзор классификация и систематизация практических аспектов взаимодействия в крупных IT-проектах, с последующей выработкой рекомендаций по внедрению в учебный процесс, изучения программных решений, используемых для организации деятельности разработчиков в процессе создания IT-продуктов.

В обзоре акцент был сделан на использование в учебном процессе свободного программного обеспечения (ПО). Такой подход позволяет использовать передовые программные продукты и решения, поскольку большая часть инструментов и решений IT-индустрии была выработана в процессе разработки средств открытого ПО.

Кроме того, использование открытого ПО в учебном процессе, позволяет избавить учебные заведения от необходимости закупать специализированное ПО, что в текущих условиях немаловажно для вузов любого уровня.

Особенности информационных систем с открытым ПО. Преимущества использования открытого ПО в образовательном процессе. Как было сказано выше, большинство инструментов, программной и информационной инженерии, составляющих инфраструктурное обеспечение крупных IT-проектов, было выработано под влиянием решений, найденных в ходе создания ПО с открытым исходным кодом (OS GNU Linux, веб-сервер Apache, IDE Eclipse и Emacs, GNU GIMP, CAS Maxima, web-браузеры и почтовые клиенты семейства Mozilla, офисные пакеты Open/Libre Office). При этом принципы, сложившиеся в процессе создания и сопровождения свободного ПО, в настоящее время распространились как на открытые, так и на проприетарные, коммерческие и корпоративные проекты IT-индустрии. Разработка открытого ПО изначально предполагала взаимодействие независимых специалистов, распределенных в пространстве и времени. Можно говорить о следующих принципиальных отличиях, выделяющих процесс создания и сопровождения продуктов с открытым исходным кодом от остальных систем [4 – 5]:

- принципиальная доступность и публичность текстов исходных кодов ПО (тексты программного обеспечения доступны через Интернет, и их изучение поощряется разработчиком);
- симбиоз пользователей и разработчиков (часто разработчик сам является активным пользователем создаваемого им ПО, а пользователи могут участвовать в разработке ПО как путем непосредственного исправления исходных кодов, так и выдвигая новые требования к сопровождаемому ПО и взаимодействуя с разработчиками);
- социальный контракт (многие сообщества свободного ПО организованы по децентрализованной схеме и опираются на организованные сообщества разработчиков и пользователей);
- возможность выбора ролей в разработке (благодаря децентрализации разработки ПО и большому количеству активных разработчиков для отдельного участника процесса появляется возможность выбора своей роли в процессе создания ПО);
- географическая распределенность участников процесса создания свободного ПО (в большинстве проектов открытого ПО разработчики распределены по разным географическим регионам, что делает критически важным создание инфраструктурной

платформы для разработки – в противном случае разработчики просто не смогут взаимодействовать).

В настоящее время во всем мире открытое и свободное ПО широко внедряется в образовательный процесс. В частности, использование такого ПО позволяет строить образовательный процесс на принципах социального и образовательного конструктивизма, согласно которым эффективный подход к обучению предполагает максимальную вовлеченность обучаемого в активную деятельность – в данном случае в работу с реальными исходными текстами существующего ПО, участие в реальных проектах разработки ПО и так далее [6 – 7]. Использование открытого и свободного ПО в процессе изучения дисциплин программной и информационной инженерии имеет следующие преимущества:

- возможность работы с настоящим кодом вместо учебных примеров;
- обучение путем изучения исходных текстов чужих программ;
- участие в реальных процессах разработки и сопровождения ПО;
- получение опыта в разработке и сопровождении ПО еще до окончания учебного курса;
- оценка вклада каждого из обучаемых непосредственно разработчиками сообщества (публикуемый обучаемым код оценивается сообществом разработчиков по схеме, аналогичной рецензированию в научных журналах);
- возможность работы не только в качестве разработчика, но и в качестве исследователя (множество проектов открытого ПО имеют значимую «гуманитарную составляющую» – экологические, медицинские, энергосберегающие проекты и так далее);
- снижение затрат на организацию обучения (большинство открытого ПО является бесплатным и позволяет использование результатов по открытыми лицензиями различных типов).

Добавим, что перечисленные преимущества дополняются тем, что в большинстве крупных IT-компаний (в том числе связанных с аутсорсингом) опыт участия в проектах, связанных со свободным ПО, рассматривается как весомое конкурентное преимущество при приеме на работу [8].

Обзор и классификация видов взаимодействия в крупных IT-проектах. Назовем комплекс концепций, знаний, навыков и инструментов, обеспечивающих коллективное взаимодействие групп субъектов при создании IT-продукта, «инфраструктурным обеспечением IT-проекта» либо просто – «инфраструктурой IT-проекта» (ИП).

Процесс разработки программного обеспечения включает в себя следующие составляющие:

- управление изменениями;
- управление и коммуникация в проекте;
- регистрация ошибок и проблем;
- целеполагание;
- управление знаниями;
- управление требованием и тестированием.

В работе [9] выделяется три основных класса информационных сущностей, которые задействованы в проекте:

- собственно сам продукт;
- проблемы и цели, связанные с продуктом;
- знания, связанные с продуктом.

Этим составляющим соответствуют три основных типа программных решений, а именно:

- системы контроля версий;
- багтрекинговые системы;
- системы авто- и самодокументирования.

Системы контроля версий (СКВ) обеспечивают совместный, распределенный в пространстве и времени доступ к исходным кодам продукта, а также хранение истории процесса разработки ПО, возможность ветвления версий ПО (основные и неосновные релизы, версии с различным функционалом и для различных программных и аппаратных платформ). «Побочным эффектом» работы СКВ является резервирование проекта: по свидетельству разработчиков, в проектах, использующих СКВ, не было случаев безвозвратной потери кода на стадии разработки и тестирования [3].

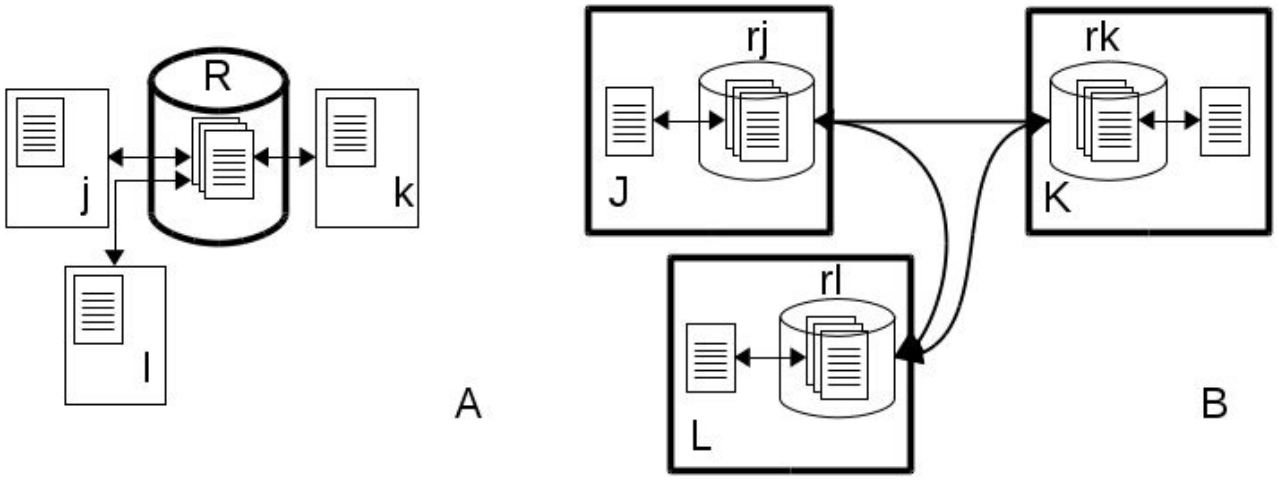

Рис. 1. Структурные схемы взаимодействия разработчиков при использовании централизованной (A) и децентрализованной (B) СКВ

Среди современных СКВ можно выделить две основных разновидности – централизованные СКВ и распределенные СКВ. Они отличаются местом размещения хранилища версий, которое в терминологии СКВ называется «репозиторием». Примеры организации взаимодействия между разработчиками в среде централизованной и децентрализованной СКВ приведены на рис. 1. В случае A взаимодействие между рабочими копиями разработчиков j, k, l происходит через центральный репозиторий R. В случае B у каждого из разработчиков J, K, L имеется свой локальный репозиторий (обозначены как rj, rk, rl), что позволяет каждому пользователю вести обмен данными непосредственно с локальным репозиторием и синхронизировать изменения между различными репозиториями, не влияя на основной рабочий процесс.

Наиболее популярной из централизованных СКВ является SVN (Subversion), созданная в 2000 году компанией CollabNet для замены более ранних СКВ, таких как CVS [10].

Однако во многих проектах SVN из-за своей централизации считается недостаточно гибкой и затрудняющей разработку. В настоящее время большей популярностью пользуются децентрализованные СКВ [11]. Основными представителями данного класса СКВ являются Git, Mercurial и Bazaar. Безусловным лидером среди этих систем является Git, за которым следует Mercurial. В свою очередь Bazaar из-за системных недостатков постепенно теряет позиции.

Багтрекинговые системы (БС) предназначены для организации взаимодействия разработчиков, тестеров и пользователей ПО на стадиях создания, тестирования и сопровождения ПО. Первоначально БС использовались исключительно как инструменты сбора и отслеживания ошибок в ПО, однако со временем их функционал и круг решаемых задач значительно расширился. В настоящее время БС используются не только для обработки информации об ошибках ПО, но и в качестве инструмента разработки, планирования и распределения задач в коллективе IT-проекта. Большинство современных БС предполагают интеграцию с СКВ и развертывание на базе web-интерфейсов, что существенно расширяет область их применения.

Лидером среди БС является Bugzilla, которую широко применяют в таких организациях, как NASA, Id Software, IBM, Novell, а также в свободных проектах Mozilla Firefox, Linux, GNOME, KDE, Apache Project, OpenOffice.org. Кроме того, популярностью пользуются продукты Mantis, GNATS, Trac.

Системы само- и автодокументирования (САД) предназначены для ведения документации и сбора информации о проекте. Среди САД выделяются две основные категории. К первой принадлежат системы автодокументирования, основанные на принципе автоматического сбора информации в ком-

ментариях исходных текстов ПО и последующей их обработке. Часто такие системы предполагают использование микроформатов или специализированных языков форматирования (markdown, textile, RST). Представителями подобных систем являются Doxygen, Sphinx, Epydoc.

Ко второй категории относятся системы самодокументирования, чаще всего использующие, так называемые вики-системы сбора информации. Преимуществами вики-систем по сравнению с традиционными системами создания и ведения документации, являются универсальность, доступность (за счет web-интерфейса, доступ может быть организован как в локальной, так и в глобальной информационной сети в реальном времени), расширяемость (в документацию могут быть включены иллюстрации, диаграммы и мультимедийные файлы), оперативность и интуитивно-понятный для неспециалистов интерфейс. Последний фактор является значимым, поскольку в создании IT-продукта участвуют различные специалисты, которым необходим единый простой способ взаимодействия с системой сбора и ведения документации.

Среди вики-систем наиболее распространенным и разработанным является движок MediaWiki, используемый фондом «Викимедиа» для Википедии. Из признанных лидеров стоит назвать MoinMoin, написанный на языке Python, и DokuWiki, который в качестве базовых единиц хранения использует текстовые файлы.

Отдельно стоит отметить продукт Fossil, представляющий собой интегрированный инструмент, который сочетает в себе черты распределенной СКВ, БС и САД.

Для эффективной работы необходимо подбирать такой набор ПО, который может взаимодействовать между собой, например, багтрекер должен позволять ссылаться на определенную версию изменений в СКВ. Наиболее удобным вариантом является объединение ПО ИП на базе веб-платформы. Такое интегрированное решение (веб-сервер с развернутыми на нем СКВ, БС и КАС) обеспечивает простой и удобный доступ через Интернет при распределенной и дистанционной разработке IT-продукта.

Большинство упомянутых в обзоре систем включает в себя поддержку веб-технологий. Например, Mercurial имеет как встроенные веб-средства (небольшой веб-сервер с документацией и автоматически генерируемыми отчетами о состоянии репозитория), так и средства интеграции, в современные веб-серверы и взаимодействия по протоколам https и ssh.

Современные сетевые сервисы, предоставляющие услуги хостинга программных продуктов (GitHub, Bitbucket, SourceForge), обеспечивают пользователю инфраструктуру, которая включает все три вида описанных в обзоре программных решений [9]. Например, Bitbucket предоставляет СКВ (Mercurial либо Git), багтрекинговую систему (Issue Tracker в терминологии платформы), а также систему автодокументирования, использующую Creole-диалект вики-движка.

Изучение практических аспектов технологий ИП. Подготовка IT-специалистов в вузе, на наш взгляд, должна включать в себя изучение всех перечисленных выше практических аспектов функционирования технологий ИП.

Как правило, изучение средств ИП зависит от имеющегося в наличии программного и аппаратного обеспечения. Здесь проявляется еще одно преимущество использования в образовательном процессе открытого ПО. В большинстве случаев такое ПО является кроссплатформенным, то есть существует в версиях для различных операционных систем. Кроме того, системы с открытым кодом нетребовательны к аппаратным ресурсам. Все перечисленное обеспечивает гибкость, позволяющую организовать процесс обучения на большинстве существующих парков вычислительных систем.

Из перечисленных разновидностей ПО наиболее сложными для понимания и изучения являются СКВ [12], [13], поэтому, на наш взгляд, основное внимание следует уделять этому виду ПО и способам его развертывания и интеграции в прочие средства ИП.

Изучение начальных принципов и задач СКВ мы рекомендуем начинать с рассмотрения СКВ SVN. Эта СКВ:

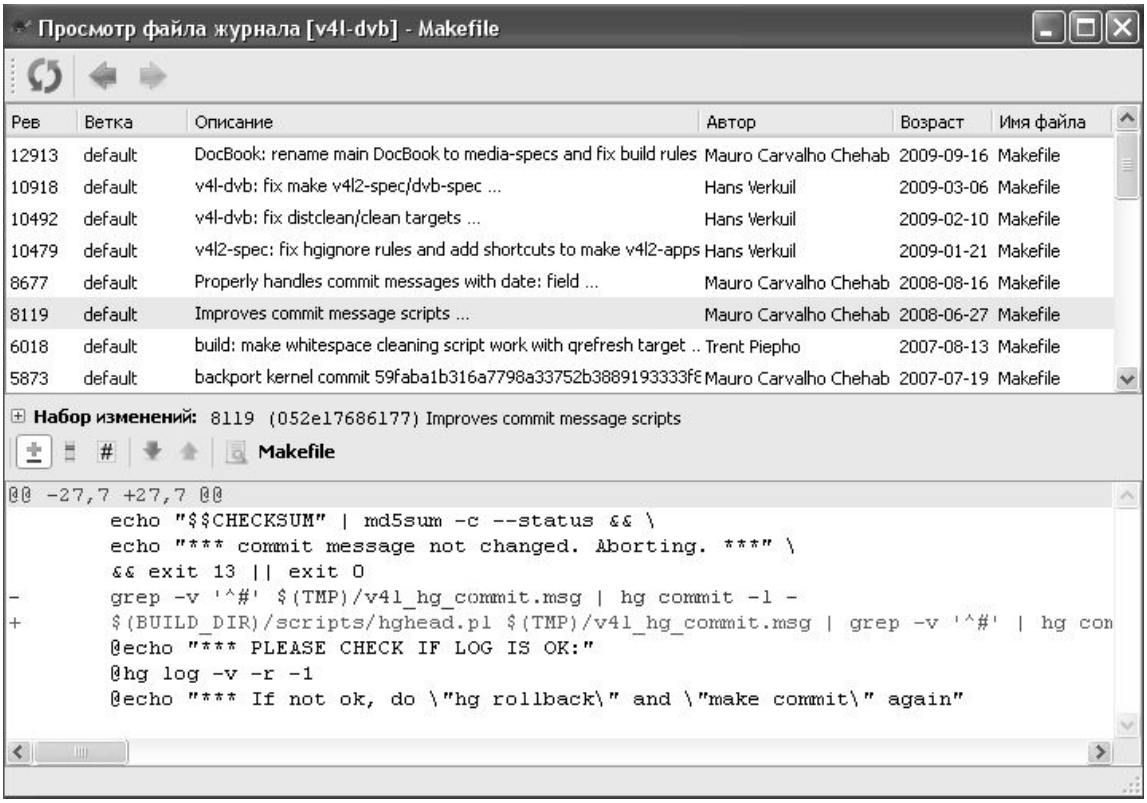

Рис. 2. Графический интерфейс TortoiseHG

1) является промышленным стандартом и распространена в индустрии почти так же широко, как наиболее популярная из распределенных СКВ;

2) снабжена хорошо переведенной русской документацией;

3) благодаря централизации позволяет рассматривать процесс взаимодействия атомарных транзакций на уровне «репозиторий – рабочая копия» на простых примерах;

4) кроме интерфейса командной строки обладает несколькими графическими надстройками, в частности TortoiseSVN, что упрощает и делает наглядной работу с СКВ в среде OS семейства Windows.

После изучения основ SVN следует перейти к изучению СКВ децентрализованного типа. Хотя лидером среди распределенных СКВ является Git, изучение таких СКВ, на наш взгляд, следует начинать с Mercurial. Принципы построения этой СКВ проще для понимания, поскольку Mercurial разрабатывалась как система, призванная заменить SVN. Например, работа с локальным репозиторием в Mercurial строится на принципах, аналогичных взаимодействию «рабочая копия – центральный репозиторий» в SVN. Еще одним преимуществом Mercurial является хорошая адаптация СКВ для среды OS Windows, которая в настоящее время является одной из самых распространенных операционных систем в учебных заведениях. Особенно полезно наличие надстройки графического интерфейса TortoiseHG, которая по своим свойствам и назначению аналогична TortoiseSVN. Пример интерфейса TortoiseHG с фрагментом репозитория одного из модулей видеодрайвера ядра Linux (http://linuxtv.org) приведен на рис. 2. С рабочей версией репозитория можно ознакомиться по адресу: http://linuxtv.org/hg/v4l-dvb.

Изучение перечисленных выше типов ПО ИП может производиться на индивидуальном рабочем месте, однако наиболее эффективным является изучение их взаимодействия в единой системе – в стеке ПО. Оптимальным вариантом будет развертывание локального вебсервера и установка на нем всего стека ПО ИП, перечисленного выше. Учебный набор может включать в себя следующие продукты: SVN, Mercurial, Git, Bugizilla, WikiMedia. Такое объединение ПО ИП на платформе вебсервера позволит развернуть на сравнительно скромных аппаратных мощностях полноценную инфраструктуру IT-проекта.

Эксплуатация такого серверного решения позволит усвоить основные принципы взаи-

модействия разработки и поддержки IT-продуктов. Пользователи подобного проекта быстро находят эффективные практики коллективной IT-разработки, такие как: хранение исходных кодов вместо бинарных представлений, возможность «сборки» проекта «одной кнопкой» и так далее.

Важно уделять внимание разработке не только ПО, но и других информационных продуктов: сайтов, информационных систем, фреймворков, документации к ПО (в форматах XML, ReST и Markdown), баз данных и отчетов исследовательских проектов (в форматах диалектов LaTeX). При изучении организации IT-сервиса будут полезными материалы предыдущих учебных курсов (например, совместная разработка программного обеспечения на базе CASE-инструментов различного рода, gcc, GNU Make и GDB).

Следует учитывать, что перечисленные выше сетевые сервисы (GitHub, Bitbucket, SourceForge) имеют бесплатную либо условно-бесплатную систему регистрации, что при наличии Интернета обеспечивает их доступность для учебной практики. Это позволяет использовать такие сервисы как для ознакомления студентов с принципами работы существующих IT-проектов свободного ПО, так и для реализации собственных проектов.

Выводы. Понимание назначения и принципов работы и знание перечисленных инструментов ИП является необходимым для участников и руководителей современных крупных IT-проектов. Владение этим инструментарием значительно облегчает как адаптацию участника в существующем коллективе крупного IT-проекта, так и самостоятельное развертывание IT-проектов.

Освоение инструментов стека ПО ИП позволяет гибко организовывать учебные комплексы, адаптируясь к существующему аппаратно-программному обеспечению, и должно строиться по принципу «от простого к сложному» (отдельный компьютер с набором ПО, сервер в локальной сети, сетевой сервис большого IT-проекта). Наиболее подходящим для учебного процесса является использование стека ПО ИП с открытым исходным кодом, позволяющего пройти все этапы изучения — от базовых навыков до освоения сложных существующих систем — с минимальными вложениями в организацию проекта.

Список использованной литературы

1. Україна. Верховна Рада. Рекомендації парламентських слухань на тему: «Створення в Україні сприятливих умов для розвитку індустрії програмного забезпечення» / Відомості Верховної Ради України (ВВР) – 2012. – № 42 – Ст. 548.

2. Брукс Ф. П.. Проектирование процесса проектирования: записки компьютерного эксперта = The Design of Design: Essays from a Computer Scientist. – М. : «Вильямс», 2012. – 464 с. – ISBN 978-5-8459-1792-8.

3. Спольски Дж. Х. Лучшие примеры разработки ПО. – СПб. : Питер, 2007. – 208 с.: ил.

4. Реймонд Эрик С. Искусство программирования для Unix: Пер. с англ. – М. : Издательский дом «Вильямс», 2005. – 544 с.: ил. – Парал. тит. англ.

5. Chavez C. et al., (2015), Free/Libre/Open Source Software Development in Software Engineering Education: Opportunities and Experiences, url: http://fees.inf.puc-io.br/FEESArtigos/artigos/artigos_FEES11/fees11_02.pdf (accessed 31.03.2015)

6. Kamthan P., (2000), On the Prospects and Concerns of Integrating Open Source Software Environment in Software Engineering Education, *Journal of Information Technology Education: Research – science Education in the 21st Century*. Springer Science & Business Media, 273 p., ISBN 978-1-4612-7084-3

7. Greening T. (ed.)., (2000), Computer Science Education in the 21st Century, *Springer Science & Business Media*, 273 p., ISBN 978-1-4612-7084-3

8. Long J., (2009), Open Source Software Development Experiences on the Students' Resumes: Do They Count?-Insights from the Employers' Perspectives, *Journal of Information Technology Education: Research*, Vol. 8, No. 1, pp. 229 – 242.

9. Фомин С. Интеграция систем управления разработкой // Открытые системы. СУБД. – 2008. – №. 2. – С. 40 – 45.

10. Glassy L., (2006), Using Version Control to Observe Student Software Development Proc-

esses, *Journal of Computing Sciences in Colleges*, Vol. 21, No. 3, pp. 99 – 106.

11. Mockus A., (2009), Amassing and Indexing a Large Sample of Version Control Systems: Towards the Census of Public Source Code History, *Mining Software Repositories, MSR'09. 6th IEEE International Working Conference on. – IEEE*, pp. 11 – 20.

12. Rocco D., and Lloyd W., (2011). Distributed Version Control in the Classroom, *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. – ACM*, pp. 637 – 642.

13. Reid K.L., and Wilson G.V., (2005), Learning by Doing: Introducing Version Control as a Way to Manage Student Assignments, *ACM SIGCSE Bulletin*, Vol. 37, No. 1, pp. 272 –276.

Получено 19.03.2015

References

1. Ukrai'na. Verhovna Rada. Rekomendacii' parlaments'kyh sluhan' na temu: «Stvorennja v Ukrai'ni spryjatlyvyh umov dlja rozvytku industrii' programnogo zabezpechennja» [Providing Favourable Environment for Software Development Industry in Ukraine], (2012), *Vidomosti Verhovnoi' Rady Ukrai'ny (VVR)*, No. 42, 548 p. (In Ukrainian).

2. Frederik P., (2012), Bruks Proektirovanie protsessa proektirovaniya: zapiski komp'yuternogo eksperta [The Design of Design: Essays from a Computer Scientist], Moscow, Russian Federation, *Vil'yams*, 464 p., ISBN 978-5-8459-1792-8 (In Russian).

3. Spol'ski D zh., (2007), Kh. Luchshie primery razrabotki PO [The Best Software Writing], SPb.: Piter, Russian Federation, 208 p.: il. (In Russian).

4. Reimond, and Erik S. Iskusstvo programmirovaniya dlya Unix [The Art of Unix Programming], (2005), per. s angl., Moscow, Russian Federation, *Izdatel'skii dom Vil'yams*, 544 p.: il., *Paral. tit. angl*. (In Russian).

5. Chavez C. et al., (2015), Free/Libre/Open Source Software Development in Software Engineering Education: Opportunities and Experiences (In English)
url: http://fees.inf.puc-io.br/FEESArtigos/artigos/artigos_FEES11/fees11_02.pdf (accessed 31.03.2015).

6. Kamthan P., (2007), On the Prospects and Concerns of Integrating Open Source Software Environment in Software Engineering Education, *Journal of Information Technology Education: Research*, Vol. 6, No. 1, pp. 45 –64 (In English).

7. Greening T. (ed.), (2000), Computer Science Education in the 21st Century, *Springer Science & Business Media*, 273 p.,ISBN 978-1-4612-7084-3 (In English).

8. Long J., (2009), Open Source Software Development Experiences on the Students' Resumes: Do They Count?-Insights from the Employers' Perspectives, *Journal of Information Technology Education: Research*, Vol. 8, No. 1, pp. 229 – 242 (In English).

9. Fomin S. Integratsiya sistem upravleniya razrabotkoi [Development Management Systems Integration], (200), Otkrytye Systemic. SUBD, No. 2, pp. 40 – 45 (In Russian).

10. Glassy L., (2006), Using Version Control to Observe Student Software Development Processes, *Journal of Computing Sciences in Colleges*, Vol. 21, No. 3, pp. 99 – 106 (In English).

11. Mockus A., (2009), Amassing and Indexing a Large Sample of Version Control Systems: Towards the Census of Public Source Code History, *Mining Software Repositories,. MSR'09, 6th IEEE International Working Conference on., IEEE*, pp. 11 – 20 (In English).

12. Rocco D., and Lloyd W., (2011), Distributed Version Control in the Classroom, *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. – ACM*, pp. 637 – 642 (In English).

13. Reid K.L., and Wilson G.V., (2005), Learning by Doing: Introducing Version Control as a Way to Manage Student Assignments, *ACM SIGCSE Bulletin*, Vol. 37, No. 1, pp. 272 – 276 (In English).

Бойко
Виктор Дмитриевич,
канд. техн. наук,
доцент каф. информационные технологии
Одесского нац. морского ун-таю
E-mail:
boyko-work@yandex.ru