Роберт Тинкер

ОБРАЗОВАТЕЛЬНЫЕ ПРОГРАММЫ С ОТКРЫТОМ КОДОМ[1]

Введение

Нынешнее образование все еще не в полной мере использует возможности персональных компьютеров. Современные технологии позволяют создавать цифровые образовательные ресурсы, которые могли бы существенно повысить эффективность образовательного процесса. Однако сегодня существует все еще слишком мало приложений, способных реализовывать преимущества тех компьютеров, что стоят у школьников на партах и легко помещаются в их портфелях, а по своей вычислительной мощности приближаются к тому, что мы еще недавно называли суперкомпьютерами. Сегодня ограничение состоит не в недостаточной мощности школьных компьютеров, а в низком качестве цифровых ресурсов. Хорошие программы стоят дорого, а средств на их закупку и поддержание у школ явно недостаточно. Решение состоит в том, чтобы кардинально изменить процесс разработки и внедрения цифровых ресурсов, перейдя к программному обеспечению с *открытым исходным кодом (open source).* В тексте мы будем писать короче, — *с открытым кодом*.

В этой статье описывается, как педагоги и программисты во всем мире могли бы создать общую библиотеку образовательных приложений, основанных на ультрасовременном образовательном программном обеспечении с открытым кодом, что помогло бы реализовать потенциальные возможности новых технологий для преображения образования. Достаточно большое количество свободно распространяемых образовательных приложений с открытым кодом и поддержка со стороны сообщества сподвижников может привести нас к формированию новых образовательных моделей — устойчивых и высокоэффективных.

История GNU/Linux — впечатляющая демонстрация того, как стратегия открытых кодов приводит к качественному коду, который является свободно-распространяемым и все-таки может быть использован для создания коммерческого ПО и коммерческой системы поддержки этого ПО. Эта статья, однако, не о системе Linux и не о роли этой системы в образовании. Мы будем говорить об образовательных *приложениях* с открытым кодом, которые выполняются в любой операционной системе, включая Windows, Apple OS и Linux.

[1] *Bob Tinker.* Open source educational applications. Перевод с англ. Андрея Маргулева.

Известны разнообразные модели применения компьютеров в образовании. Одно из наиболее перспективных направлений основано на мощных инструментальных средствах и моделях, которые можно использовать во многих сферах обучения (*Feurzeig и Roberts*, 1999; *Gobert и Buckley*, 2000; *Tinker*, 1990a, 1990b).

В рамках этого направления учащиеся учатся, решая, с помощью этих инструментов и моделей, задачи. При этом вся учебная деятельность идет в единой информационной среде, включенной в Интернет, фиксирующей ход учебного процесса и позволяющей различными способами оценивать этот ход. Педагоги при этом могут перенастраивать содержание занятий и систему оценивания так, чтобы приспособить стратегии обучения к потребностям и интересам своих учащихся.

Инструментальные средства, модели и платформы, необходимые при таком подходе, сложны и даже, вероятно, слишком сложны, чтобы их разрабатывать на коммерческой основе. Лучший выход — это производить их с открытым кодом и верить, что их оценит широкое сообщество пользователей, которые обеспечат дальнейшую поддержку и развитие.

Программное обеспечение с открытым кодом

Программное обеспечение с открытым кодом совершило переворот в создании, обслуживании и распространении больших программных пакетов. Понятие программного обеспечения с открытым кодом определяется уникальным условием имущественных авторских прав («копирайта»), свидетельствующим, что любой может использовать исходный текст программы бесплатно, но, изменяя его, должен выпустить измененный пакет с тем же самым условием копирайта. Это означает, что исходные коды всегда будут доступны и свободно распространяемы. Нет полного согласия относительно «самой лучшей» формулировки имущественного авторского права для ПО с открытым кодом, но сообщество в поддержку открытого кода создало некоммерческую организацию для проверки различных лицензий. Эта организация — OpenSource.org поддерживает определение открытого кода, утвердив список из более чем десяти видов лицензий на продукты с открытым исходным кодом[2]. Исходный код, выпущенный под каждой из этих лицензий, имеет полное право называться «открытым исходным текстом».

Краткая история Linux

Для понимания значимости лицензии открытого кода и связанных с этой идеей возможностей улучшения образования, важно знать ее историю и эволюцию[3]. История эта во многом связана с операционной системой Linux.

В 1980-х годах на университетских компьютерах почти повсеместно использовалась операционная система Unix. Сначала Unix

[2] http://www.opensource.org/licenses/

[3] Лучшая история открытого кода — «Open Sources: Voices from the Open Source Revolution» («Открытые исходные тексты: голоса революции открытого кода») в свободном сетевом доступе — http://www.oreilly.com/catalog/opensources/book/toc.html

была защищена авторским правом AT&T, но распространялась свободно среди университетских пользователей. Затем AT&T начал брать плату с университетов за пользование Unix, и многие программисты почувствовали себя обманутыми, потому что они должны были платить за доступ к текстам Unix, ряд которых они сами написали, бесплатно внеся свой вклад в рост сообщества пользователей Unix. В ответ Ричард Столмен организовал проект GNU (означает — GNU's Not Unix... — *Прим. пер.*) — проект для разработки системных компонентов с открытыми исходными текстами, расширяющих ядро операционной системы Unix.

Линус Торвальдс отделил открытый код от Unix, написав ядро операционной системы, названное Linux и сделав открытым его исходный текст. Linux дублирует функции и интерфейс GNU-компонентов Unix. Окончательная комбинация GNU/Linux и дополнительного открытого кода стала широко известна просто как операционная система Linux. Со времени своего происхождения в качестве реакции на корпоративную политику она выросла в одну из самых популярных операционных систем в мире. По иронии судьбы, GNU/ Linux теперь поддерживается большими корпорациями. Например, IBM поставляет систему GNU/Linux на своих компьютерах и инвестировала миллиарды долларов в программное обеспечение и сервисы на ее базе![4] Hewlett-Packard, Intel, IBM, NEC и Oracle — это лишь несколько крупных компаний, использующих теперь Linux. Эти компании платят своим собственным программистам за развитие открытого кода и вносят свой вклад в различные совместные проекты с открытым кодом, координирующие усилия добровольцев по его развитию.

Своей феноменальной популярностью система GNU/Linux обязана не только тому, что является свободно распространяемой, но и тому, что эта система лучше большинства альтернатив: она поддерживает более широкий набор аппаратных средств, меньше по объему, более надежна[5], более легко расширяема, более функциональна и быстрее, чем другие версии Unix, а тем паче, чем Windows. Операционная система OS X для компьютеров Макинтош основана на ОС Free BSD.

Наличие у GNU/Linux открытых кодов позволяет привлечь к работе по развитию системы больше способных людей, чем может нанять Microsoft. В результате возникшее для решения общих задач международное сообщество пользователей-разработчиков совершенствует программное обеспечение и проводит критический анализ ведущихся работ. Коды, которые при этом выживают, имеет тенденцию быть лаконичными, элегантными и надежными. Благодаря открытому коду программы становятся общественным достоянием и все более качественными.

Эта краткая история GNU/Linux дает уроки того, как можно производить программное обеспечение с открытым кодом. Возможно,

[4] IBM Annual Report, 2000. P. 30.
[5] См. http://www.gnu.org/software/reliability.html.

те механизмы, которые произвели GNU/Linux, смогут привести к созданию и лучших образовательных инструментов, чем те, которые мы имеем сегодня.

Прикладные инструментальные системы с открытым кодом

Лидеры движения за открытый код, такие как Столмен (2002), Торвальдс (*Torvalds & Diamond*, 2002), настаивают, что открытый код — лучший способ разработать любой вид надежного и элегантного программного обеспечения, и предсказывают, что большая часть программного обеспечения будет рано или поздно распространяться с открытым исходным кодом. Идея открытого кода вышла далеко за пределы операционных систем. В дополнение к нескольким браузерам теперь есть и электронные таблицы, и текстовые редакторы, и презентационный пакет (подобный PowerPoint), и gimp — программа обработки изображений, сравнимая с Photoshop, все — с открытым кодом.

Важно понимать, что приложения с открытым кодом не обязаны работать под операционной системой Linux. Школы могут принять политику использования приложений с открытым кодом без перехода на Linux. Они могут использовать и приложения не с открытым кодом, существенным здесь оказывается и другой фактор — многоплатформенность. В частности, самое новое образовательное программное обеспечение пишется в Java, потому что это лучшая многоплатформенная среда. Программное обеспечение, написанное в Java, может работать под операционными системами Windows, Mac, Linux и др. Школы, с их разнообразными, часто старыми, компьютерами и операционными системами, хотят использовать приложения, которые работали бы под любой операционной системой, и в большинстве случаев приложения, написанные в Java, будут удовлетворять этому требованию[6].

Имеется важное различие между ПО с открытым кодом и ПО в общественном достоянии. ПО в общественном достоянии не имеет копирайта и может быть просто исполнимым модулем приложения, а не его исходным текстом. И если источник недоступен, то сообщество не имеет никакой возможности развивать его чтобы улучшить. Но даже если источник приложения в общественном достоянии доступен, тот, кто вносит даже незначительные усовершенствования, может всегда поставить на них копирайт. Это значит, что пользователи ПО в общественном достоянии могут в любой момент оказаться обязанными платить за него, в то время как они считали его бесплатным. Эта угроза возможной будущей потери доступа к ПО в общественном достоянии затрудняет внесение в него усовершенствований. В результате ПО в общественном достоянии остается замороженным во времени и отмирает из-за отсутствия поддержки.

ПО в общественном достоянии зачастую не поддерживается. Любое программное обеспечение требует ресурсов двух родов —

[6] Так получилось, что Java — система с открытым кодом, но это не означает, что программы, написанные в Java, должны быть с открытым кодом.

на создание и на поддержку. Роль создания очевидна, но слишком часто решающая роль поддержки не учитывается. Программное обеспечение без поддержки почти бесполезно, потому что когда ошибка вызвана изменениями в программном обеспечении, используемом приложением, этого нельзя устранить. Например, приложение могло быть написано в Java 1.4, имеющем небольшую несовместимость с Java 1.5, а теперь все компьютеры снабжаются версией 1.5. Или операционная система могла измениться, и некоторые старые вызовы могли замениться при этом более совершенными. Или некоторый изъян в приложении не был очевиден, пока компьютеры не стали работать быстрее. Список источников проблем бесконечен, и достаточно всего одной, чтобы сделать пакет бесполезным. Не поддерживаемое программное обеспечение в общем случае будет терять функциональность и, в конечном счете, перестанет работать, поскольку и ПО, и оборудование модернизируются. Это фатально для ПО в общественном достоянии, но не для ПО с открытым кодом, которое поддерживается пользовательским сообществом. Важно понимать, что открытый код — не ПО в общественном достоянии, которое не имеет никакого копирайта. Напротив, открытый код защищен авторским правом, но в таком виде, который гарантирует, что любые сделанные усовершенствования будут доступны всем, и, таким образом, ПО с открытым кодом может поддерживаться и развиваться.

Открытый код и бизнес

На первый взгляд может показаться, что программное обеспечение с открытым кодом несовместимо с коммерциализацией, и что компании будут избегать его. Однако при ближайшем рассмотрении становится ясно, что открытый код создает для предпринимателей много возможностей. Например, самый простой способ установить GNU/Linux на вашем компьютере состоит в том, чтобы купить выпуск GNU/Linux от компании, подобной Fedora[7]. Можно найти точно такой же материал бесплатным в Интернете, но время, сэкономленное при использовании автоматического инсталлятора компании, обеспечившей хорошо проверенные, совместимые, документированные, настроенные компоненты, вполне стоит небольшой оплаты. GNU/Linux создает благоприятные сервисные и аппаратные перспективы, которые оправдывают те огромные инвестиции, которые компании, подобные IBM, вкладывают в открытый код. Поэтому возможно, что образовательные приложения с открытым кодом обеспечат возможность существования для разнообразных коммерческих образовательных сервисов — например, создание комплексов приложений, простые инсталляторы, встраивание в существующие системы управления образовательным процессом, обеспечение консультативных услуг и профессиональную подготовку преподавателей, использующих ПО с открытым кодом.

[7] http://www.redhat.com/en_us/USA/fedora/.

Программное обеспечение с открытым кодом в образовании

История программного обеспечения с открытым кодом дает основания предположить, что приверженность открытому коду могла бы помочь поставлять в образование столь сложные разновидности ПО, которые коммерческие поставщики вообще не способны создавать.

Почему открытый код работает?

Идея ПО с открытым кодом поначалу кажется утопичной и нереалистичной. Преобладающий способ производства ПО дорог, жестко контролируется и требует иерархической организации, воплощенной Microsoft. Как могла группа добровольцев, не оплачиваемых хакеров, без планирования и почти без организации каким-либо образом сделать что-нибудь стоящее? Это бросает вызов логике. И что еще более важно: если хорошее ПО было произведено, то как оно будет поддерживаться? Тем не менее огромный рост объемов ПО с открытым кодом — доказательство того, что на эти вопросы был дан ответ.

Исследуя успешное ПО с открытым кодом, можно выяснить, как обеспечивались его создание и поддержка. Это дает ключ к тому, какие характеристики необходимы, чтобы создавать и поддерживать образовательное ПО с открытым кодом. Вот что удается обнаружить:

Широкое использование. Apache[8] удовлетворил потребность в Web-сервере и стал теперь самым популярным из Web-серверов в мире. В какой-то момент каждый успешный пакет с открытым кодом достигает в своей эволюции точки, в которой достаточно много людей начинают так от него зависеть, что распределенные пользователи хотят (и способны) обеспечить ему развитие и поддержку. Ключевым моментом является образование «базиса» пользователей-программистов, настолько приверженных данному пакету, что они хотят инвестировать время и иные ресурсы для поддержания его жизни. Это тем более вероятно, когда ПО достаточно полезно, чтобы удовлетворить потребности множества потенциальных пользователей.

Функциональные возможности. Наиболее успешные пакеты с открытым кодом — крупные, обеспечивающие ряд важных функциональных возможностей. Любой, кто намеревается использовать открытый код, стоит перед вопросом: использовать ли свободно распространяемый открытый код, или создать ему эквивалентный? Создавать что-то самому часто очень привлекательно: это более занятно, вы контролируете процесс написания, и ваша компания может получить авторское право на сделанное вами и получать деньги за лицензию. Альтернативный вариант сопряжен с рискованной зависимостью от сообщества неизвестных хакеров, делающих код, который вам не полностью понятен. Однако экономия, проистекающая из использования открытых кодов, может перевесить подобные неудобства. Если открытый код обладает достаточной

[8] http://www.apache.org/

мощностью, чтобы существенно сэкономить время дорогостоящих программистов, то и решение о его использовании вместо создания нового ПО принимается легче.

Продуманный дизайн. Новый пользователь ПО с открытым кодом должен быть в состоянии его поддерживать и, в некоторых ситуациях, изменять. Это означает, что программисты должны хорошо понимать код продукта. Хорошо спроектированное ПО намного проще для понимания и поддержки, потому что оно документировано или самодокументировано, логично и компактно. Авторы GNU/Linux гордились качеством своего дизайна. Хороший дизайн, в частности, предполагает модульность; GNU/Linux, например, — большой набор модулей. Это давало возможность индивидам и маленьким группам, работающим независимо, делать важные дополнения к продукту.

Что все это значит в образовательной сфере?

Итак, успешное ПО с открытым кодом для образования должно обладать указанными характеристиками; оно должно состоять из хорошо спроектированных, высокофункциональных пакетов, способных удовлетворять разнообразным потребностям. Маленькие приложения, которые легко копируются, наподобие программки построения графиков функций, не «взлетят». Узкоспециальное ПО, даже хорошо спроектированное и высокофункциональное, потерпит неудачу с открытым кодом — из-за ограниченности группы специалистов, которые захотят обеспечивать поддержку.

Укажем уже существующий пример образовательного приложения с открытым кодом, соответствующего указанным параметрам. Этот продукт отвечает огромному потребительскому спросу, обеспечивает существенные функциональные возможности и хорошо спроектирован. OpenACS — платформа для дистанционных курсов, созданная в MIT, была преобразована в продукт Ars Digita, его разработка привлекла значительное финансирование в конце 1990-х годов, но кончилась финансовым крахом, продукт возродился в открытом коде под названием dotLRN. OpenACS является сокращенной версией dotLRN — продукта, который используется все расширяющейся группой университетов во всем мире и поддерживается сообществом компаний[8].

OpenACS решает проблему, которая стоит почти перед каждым университетом в мире и многими другими институтами, а именно: какую технологию использовать для дистанционных курсов. Мы начали предлагать такие курсы десять лет назад, сначала используя Web-страницы, затем ожидали пакета какого-либо университета, но он так и не появился. Затем мы использовали LearningSpace, который в дальнейшем IBM перестала поддерживать, и, наконец, перешли на Blackboard. Каждая из этих платформ и множество других, исследованных нами, имеют ограничения, которые сужают образовательную ценность материала, предоставляемого студентам. Только

[9] См. http://openacs.org/community/companies/

когда мы перешли на OpenACS, мы оказались в состоянии обратиться к исходным текстам программ и сделать необходимые усовершенствования. Поступая таким образом, мы делали именно то, что необходимо, чтобы помочь в создании жизнеспособного сообщества открытого кода. Мы добавляли новые функциональные возможности, которые может использовать каждый, не по великодушию, но потому что эти улучшения были в наших собственных интересах.

Урок этого примера состоит в том, что образовательные приложения с открытым кодом могут процветать, если ПО отвечает осознаваемой потребности, возникшей перед множеством пользователей. Он также иллюстрирует трудности коммерческих поставщиков в обеспечении долгосрочной поддержки больших образовательных пакетов.

Необходимость сотрудничества разных специалистов

Существует еще один аспект ПО с открытым кодом, который способен быть препятствием к созданию и поддержке образовательным сообществом открытого кода: разделение на тех, кто пишет программу, и тех, кто ее использует. При разработке систем с открытым кодом эти два сообщества совпадают. Когда словосочетание «открытый код» было синонимично «системным компонентам», программисты и пользователи были одним и тем же людьми — системными программистами, пишущими системные компоненты с открытым кодом, в которых они и их коллеги нуждались и которые они использовали. В результате люди, пишущие код, очень хорошо понимали, что именно необходимо. Когда открытый код распространился на такие приложения, как, например, общепользовательские и графические инструменты, программисты, возможно, не знали своей аудитории, но могли руководствоваться успешными коммерческими аналогами. В образовании разделение между программистами и пользователями более полное, и в результате разработчики программ могут потерять представление об образовательных нуждах.

Если вы ищете в Интернете образовательное ПО с открытым кодом, то можете удивиться числу полученных ссылок. Но при более близком рассмотрении большинство найденного относится к одному из двух видов: компоненты операционной системы и примитивные образовательные приложения, типа электронных пособий по методике flash cards или электронных журналов. Ни один из этих продуктов не может оказать существенного влияния на образование. Лучшие компоненты операционной системы могут, конечно, уменьшить затраты на компьютеры для образования. Хотя это и было бы ценно, но вопрос, как же использовать эти компьютеры, остается без ответа. Примитивное же ПО, которое просто укрепляет учителе-центристскую, авторитарную, основанную на запоминании фактов и алгоритмов образовательную парадигму, будет только препятствовать необходимым изменениям в образовании[10].

[10] Для обзора образовательных приложений с открытыми кодами, см. *Tinker,* 2005.

Образовательные приложения с открытым кодом найдут широкое применение только тогда, когда в их разработке будут участвовать специалисты двух различных сообществ: программисты и разработчики содержания. Так как очень немного людей могут находиться на переднем крае обеих областей, маловероятно, что работающие в одиночку энтузиасты создадут важное образовательное ПО с открытым кодом таким же образом, как Линус Торвальдс создал ядро Linux. Возможно, трудность сведения этих двух областей вместе и объясняет малочисленность образовательных приложений с открытым кодом.

Архитектура для образовательного открытого кода в наших разработках

Нам повезло, что мы разрабатывали ПО с открытым кодом в командах, куда входили лучшие в образовательной отрасли программисты, специалисты по содержанию образования в естественнонаучной, технологической, инженерной и математической (STEM — Science, Technology, Engineering and Mathematics) областях, специалисты по педагогическому дизайну. Через различные формы двустороннего сотрудничества и официальные центры наша работа подпитывалась информацией о наиболее интересных идеях в области педагогического и программного дизайна, с одной стороны, и содержания образования в области STEM — с другой. Результатом этой работы стал поток инновационных учебных материалов. Детальное описание этих материалов можно найти в (*Tinker,* 2004); что здесь важно: разрабатывая эти материалы, мы опирались на архитектуру, которая позволяет программистам и разработчикам контента работать независимо.

Направляемое учителем исследование моделей и инструменты

Наша архитектура была развита исходя из анализа наиболее плодотворных применений новых технологий способна в образовании. Очевидна огромная потребность в инструментальных средствах и моделях, которые учащиеся могут использовать при обучении в ходе направляемого учителем (управляемого) исследования: по всем предметам и на всех уровнях[10]. Модели и инструментальные средства — вот что питало распространение компьютеров не только в образовательной среде, но и в бизнесе, и в государственном управлении. Сложный продукт означает значительные инвестиции в программирование, но это оправдывает себя, если он получит широкое применение в различных дисциплинах и на разных образовательных уровнях. Мы уверены, что существует еще много такого сорта моделей и инструментов, ожидающих своего воплощения.

Акцент на управляемом исследовании важен для нашей архитектуры: школьники учатся лучше всего в процессе исследовательской

[11] Термины «инструментальные средства» и «модели» допускают много интерпретаций. «Инструментальные средства» используются здесь как программные приложения, которые относительно бесконтентны и дают пользователю полезный набор функций, подобно текстовым процессорам, браузерам, графическим редакторам и географическим информационным системам. «Модели» используются для интерактивного ПО, дублирующего некоторые из характеристик реальных или воображаемых систем типа имитатора полета, игры SimCity или климата Земли.

деятельности, но в то же время легко теряются без поддержки со стороны учителя. Таким образом, архитектура должна обеспечивать как общую поддержку исследовательской деятельности учащихся, так и контроль и контекстно-зависимую поддержку. Степень контроля и поддержки зависит от образовательной философии педагога и может быть установлена эмпирическим путем; архитектура может обеспечить по желанию педагога любую, степень контроля или поддержки.

Инструментальные средства, платформы и учебные активности

Для нас оказалось важным использовать архитектуру, которая разделяет программное обеспечение на *инструменты*, *платформы* и *учебные активности (упражнения, лабораторные работы и т.д.)*. *Модели и инструменты*, обеспечивают операционализацию содержания образования и могут применяться во многих образовательных контекстах. *Платформы* поддерживают инструментальные средства, позволяя автору формировать учебный контекст, связывать между собой понятия, предлагать указания к действию и подсказки и предоставлять дополнительную информацию. Платформа может использовать одно или более инструментальных средств, определять начальные условия моделирования и доступные функции инструментов. Платформа может также воспринимать данные, вводимые студентом, фиксировать его действия и оценивать его продвижение по курсу. *Учебные активности* — это те виды деятельности обучаемого, которые предполагаются автором данного учебного модуля. Например, мы разработали программу BioLogica, *инструментальное средство,* посвященное генетике, которое использует законы Менделя и позволяет обучаемому исследовать размножение на молекулярном, клеточном, индивидуальном и популяционном уровнях (*Buckley и др.,* 2004). Это инструментальное средство моделирует классическую генетику, включая генные признаки и летальные гены, а также мутации и влияние окружающей среды на популяции на протяжении сотен поколений, давая возможность моделировать генетический дрейф и появление новых видов. Более ранняя версия этого инструмента создавалась как автономное приложение, которое учащиеся могли бы исследовать. Но скоро мы обнаружили, что такое исследование оказывается слишком сложным, если только мы не снабжаем учащихся системой вопросов, тем для исследования и не убираем ненужные в конкретном исследовании возможности моделей.

Это привело к развитию *платформы* Pedagogica*,* первоначально предназначенной только для поддержки учебной деятельности в программе BioLogica и ее оценки (*Horwitz & Christie*, 1999; *Horwitz & Tinker,* 2001). Первоначально спроектированная для инструментальной среды BioLogica, она использовалась затем и для многих других сред. Pedagogica использует скриптовый (сценарный) язык — JavaScript — для управления учебной деятельностью, идущей на базе любой инструментальной среды. Каждый скрипт формирует определенный вид учебной деятельности. Было разработано более пятидесяти

различных *учебных активностей* для Pedagogica и различных комбинаций одной или более инструментальных сред. Некоторые из этих активностей — это просто страница с текстом, моделью BioLogica и управляющими элементами, позволяющими записывать все ответы студентов и в случае необходимости передавать их на центральный сервер для сохранения и последующей обработки. Другие активности — это богатые, ветвящиеся, многостраничные исследования, которые содержат многочисленные модели и инструменты, информационные источники и систему встроенного оценивания (embedded assessment). Каждая активность — это программа на JavaScript. Поскольку JavaScript — полноценный язык программирования, почти любые функциональные возможности могут быть встроены в учебные активности системы Pedagogica. В то же время активности могут создаваться только разработчиками, работающими в JavaScript.

Второй пример архитектуры «инструмент-платформа-активность» дает нам Molecular Workbench (Молекулярная мастерская) (*Berenfeld, Pallant, Tinker, Tinker & Xie*, 2004; *Pallant, Tinker,* 2004). *Инструмент* Molecular Workbench (MW) — сложная молекулярно-динамическая моделирующая среда, разработанная для образования, которая моделирует взаимодействия большого количества атомов и молекул друг с другом, а также с различными полями и электромагнитным излучением. Школьники, работающие в этой среде, приобретают глубокое понимание мира на атомном уровне, которое может помочь объяснять очень широкий круг явлений, обычно изучаемых в естественных науках, инженерии и технологии. Динамическая модель, созданная на базе MW, может управляться платформой Pedagogica, но авторы MW, чтобы расширить диапазон возможностей этой моделирующей среды, разработали чрезвычайно гибкую *платформу,* называемую MW Pages, которая не требует знания программирования. MW Pages — это специализированный текстовый редактор, который воспринимает модельные компоненты MW и обрабатывает их таким же образом, как был бы обработан текстовый символ. Устройства ввода данных, такие как кнопки и ползунки (слайдеры), и устройства вывода, такие как графопостроители, также допускаются текстовым редактором, который их логически связывает с динамическими моделями MW. Дополнительные мультимедийные компоненты также могут быть помещены на MW страницу, и страницы эти могут быть связаны между собой, создавая группы связанных между собой заданий-уроков. Созданный с помощью такого специализированного редактора уникальный образовательный продукт может поддерживать *деятельность ученика* на уроке[12]. Важно, что материал для поддержки *деятельности ученика* создается почти так же легко, как обычный текст, без необходимости приобретать программистские навыки в написании скриптов. В результате очень многие непрограммирующие учителя и ученые-специалисты разработали

[12] См. http://molo.concord.org для базы данных учебной деятельности, базированной на на MW.

за короткий срок более 200 учебных объемных уроков-заданий, которые внесены в общую, доступную всем базу данных активностей/уроков-заданий[13].

Что мы поняли?

Несколько важных уроков было извлечено из опыта разработки указанных пакетов.

В приведенных примерах разработчик инструментальной среды поставляет контент — генетику и молекулярную динамику. **Разработка инструментальной среды** требует знания научной области контента, но не требует педагогической квалификации, поскольку учебные активности формирует использующий платформу автор. Разработчик инструментальной среды может игнорировать функции, обеспечиваемые платформой, и концентрироваться на построении самой инструментальной среды или модели. Специалисты по контенту обычно включают избыточные (особенно для начинающих) возможности, параметры, детали. В нашей архитектуре разработчик инструментальной среды свободен в своей деятельности, поскольку автор учебных активностей будет в дальнейшем самостоятельно решать, какие параметры и функции сделать доступными для пользователя в любом конкретном контексте.

Разработчик платформы может не быть специалистом ни в контенте, ни в педагогике. Платформа является самым близким к операционной системе компонентом. От нее требуется обеспечение интерфейса с инструментальной средой и поддержка активностей, разрабатываемых автором. Полнофункциональная платформа должна также обрабатывать связи между различными инструментальными средами, обеспечивать регистрацию учащихся, загрузку учебной активности, постоянную включенность учащегося в учебную активность, оценку результатов, обрабатывать/хранить/преобразовывать разного рода документы (в том числе и аудио/видео) и т.д. в интерактивной среде.

Разработанные инструментальные среды и платформы удовлетворяют нашим критериям для успешных продуктов с открытым кодом. Они имеют высокий уровень функциональности и потенциально широкую применимость. Сотни учебных активностей, основанных на компонентах и платформах BioLogica и MW, десятки тысяч учащихся свидетельствуют о полезности систем.

Создание учебных активностей может быть коммерциализировано, поскольку их можно защитить обычным копирайтом. Весь собственно педагогический дизайн реализуется в учебных активностях, а они подобны документам текстового процессора: документы можно защитить авторским правом, даже если текстовой процессор — с открытым кодом. Поскольку до сих пор мы финансировались правительственными грантами, то предпочитали, чтобы создаваемые нами активности были свободно доступными в духе открытого кода,

[13] BioLogica, Pedagogica, the Molecular Workbench, MW Pages и многочисленные меньшие пакеты, используемые в нашей деятельности, — все они с открытыми исходными текстами, распространяются на условиях LGPL.

но ничто не заставляет нас поступать так же и в дальнейшем. Издатель, например, может использовать компоненты и платформу, чтобы разработать составляющие его собственность учебные активности. Разработка высококачественных активностей может быть дорогой, и они нуждаются в защите, чтобы издатель мог возместить затраты на разработку и поддержку. Такая модель желательна, поскольку расширяет число пользователей, заинтересованных в извлечении дохода, базирующихся на инструментальных средах с открытым кодом. Тем самым повышается вероятность будущей поддержки и усовершенствования системы этими пользователями.

Выводы

Разделение инструментальных сред, платформ и учебных активностей может быть решающим для создания ценных приложений с открытым кодом, которые начинают воплощать технологические надежды образования. Это разделение отражает тот факт, что для производства эффективного образовательного материала необходимы различные умения: системные программисты нужны для создания платформ, которые могут обрабатывать общие информационные потоки, возникающие в образовательном контексте, специалисты по содержанию нужны для разработки инструментальных сред и моделей, специалисты по педагогическому дизайну — для разработки учебных активностей. Разделяя таким образом усилия разработчиков, мы обеспечиваем независимую работу каждой группы. Этот разделение ответственности было важно в истории GNU/Linux и может оказаться столь же важным в образовательных приложениях.

Необходимо не так уж много инструментальных сред и платформ, но это — большие пакеты, которые требуют существенных программистских ресурсов. Эти ресурсы могут быть затрачены коллективами программистов, работающих с открытым кодом, поскольку инструментальные среды и платформы имеют очень широкое потенциальное применение. Как коммерческие, так и некоммерческие разработчики образовательного ПО могут использовать платформы и поддерживаемые ими инструментальные среды для создания огромных библиотек конкретных учебных активностей. Поскольку процесс авторской разработки столь же прост, как использование текстового процессора, то даже не имеющие технической подготовки преподаватели смогут настраивать существующие учебные активности в соответствии с требованиями конкретного учебного плана и нуждами обучаемых, извлекая еще больше пользы из этих ресурсов.

Инструментальные среды и платформы, разработанные консорциумом «Конкорд», служат важными примерами ценности такой архитектуры. Они уже обеспечили среду для создания большого количества учебных активностей, нашедших широкое применение. Мы полагаем, что они готовы к широкому распространению, в результате чего сформируются жизнеспособные сообщества открытого

кода. Существует много других инструментов и моделей, разработка которых могла бы увеличить ценность этих пакетов.

Мы продолжаем совершенствовать платформы и инструментальные среды, постоянно растет число основанных на них учебных активностей. В настоящее время мы добавляем простоту авторской разработки платформы MW Pages в платформу Pedagogica, а также новые функции к компоненту MW. Мы работаем с коллегами из Университета Калифорнии, Беркли над разработкой SAIL — оболочки для платформ, которая будет включать инструментальные средства, упрощающие разработку новых платформ.

Широкое распространение нынешнего поколения инструментальных сред и платформ, разработанных в консорциуме «Конкорд», не так важно, как та архитектура, которую они представляют. Развитые инструментальные средства и модели, а также возможности для обучаемых вести на их базе управляемое исследование, важны, чтобы осознать те изменения, которые делает возможными нынешняя технология. Вряд ли дальнейшая разработка таких больших образовательных приложений будет поддержана правительством или частным сектором. Единственная альтернатива — это модели разработки с открытым кодом и разделение программистской работы на небольшие задачи так, чтобы отдельные специалисты и маленькие группы могли сделать свой вклад. Одним из разумных подходов к этому выглядит разделение работ по разработке платформ, инструментальных сред и учебных активностей.

Генетика, атомные взаимодействия и много других явлений — одни и те же в каждой стране. Школьники и студенты всюду учатся лучше всего в ходе управляемого исследования. Поэтому есть глобальная потребность в различного вида продуктах и их компонентах, описанная в данной статье. Если различные народы смогут внести свой вклад в общие усилия по их разработке и поддержке, мы реализуем образовательные перспективы новых технологий намного скорее, с намного лучшим программным обеспечением — свободно распространяемым, с открытым кодом.

Литература

1. B*erenfeld, B., Pallant, A., Tinker, B., Tinker, R., & Xie, Q. C.* (2004). From genetic code to protein shape using dynamic modeling. Paper presented at the Proceedings of the NARST 2004 annual meeting, April 1–3, Vancouver, BC, Canada.

2. *Buckley, B. B., Gobert, J. D., Kindfield, A. C. H., Horwitz, P., Tinker, R., Gerlits, B., et al.* (2004). Model-Based Teaching and Learning with BioLogica: What Do They Learn? How Do They Learn? How Do We Know? // Journal of Science Education and Technology, 13(1), 23–41.

3. *Feurzeig, W., & Roberts, N.* (eds.) (1999). *Modeling and Simulation in Science and Mathematics Education.* New York: Springer.

4. *Gobert, J., & Buckley, B.* (2000). Introduction to model-based teaching and learning in science education // International Journal of Science Education, 22(9), 891–894.

5. *Horwitz, P., & Christie, M.* (1999). Hypermodels: Embedding curriculum and assessment in computer-based manipulatives // Journal of Education, 181(2), 1–23.
6. Horwitz, P., & Tinker, R. (2001). Pedagogica to the rescue: A short history of hypermodels. @*CONCORD, 5*(1), 1, 12–13.
7. *Pallant, A., & Tinker, R.* (2004). Reasoning with atomic-scale molecular dynamic models // Journal of Science Education and Technology, 13(1), 51–66.
8. *Stallman, R. M.* (2002). Free software, free society: Selected Essays of Richard M. Stallman. Boston: The Free Software Foundation.
9. *Tinker, R.* (1990a). Computer Based Tools: Rhyme and Reason. In: *E. F. Redish & J. S. Risley* (eds.). Computers in Physics Education. Reading, MA: Addison-Wesley.
10. *Tinker, R.* (1990b). Modelling and theory building: Technology in support of student theorizing. In: D. L. Ferguson (ed.). Advanced Educational Technologies of Mathematics and Science (Vol. 107. P. 91–114). Berlin: Springer-Verlag.
11. *Tinker, R.* (2004). Free computer-based learning resources. @*CONCORD, 8*(2), 1, 4–5.
12. *Tinker, R.* (2005). Freeing Educational Applications. @*CONCORD, 9*(1), 10–11.
13. *Torvalds, L., & Diamond, D.* (2002). *Just for fun: The story of an accidental revolutionary.* New York: HarperCollins Publishers. Русский перевод: *Торвальдс Л., Даймонд Д.* Ради удовольствия. М.: ЭКСМО-Пресс, 2002.