Визитка

ДЕНИС СИЛАКОВ, к. ф.-м. н., член рабочей группы LSB, старший архитектор ЗАО «РОСА», занимается автоматизацией разработки ОС «РОСА»

Участие в открытых проектах
как начало профессиональной карьеры

Интересные вакансии требуют опыта, который можно получить, лишь поработав на аналогичном месте… Узнайте, как свободное и открытое ПО помогает разорвать этот порочный круг!

При отборе кандидатов на ту или иную вакансию производится оценка способности человека к выполнению соответствующих обязанностей. Комплекты навыков, которыми должен обладать человек для решения определенного класса профессиональных задач, часто называют компетенциями.

Каждая компания имеет свои задачи, свои требования к кандидатам и собственную модель компетенций. Однако в пределах одного сектора экономики эти модели достаточно сильно пересекаются. В частности, можно сказать, что ИТ-профессионал должен обладать компетенциями трех видов:

- техническими (например, владеть различными методами и техниками разработки программ, их отладки, настройки и использования);
- когнитивными (включающими понимание требований и ограничений в рамках предметной области, умение отделять существенное от несущественного для конкретной задачи, упорядочивание задач по их приоритетности, построение решений на основе неполной информации);
- социальными (подразумевается эффективное общение с другими людьми, умение работать в команде и координировать свою работу с нуждами ее членов, аргументация точки зрения, адекватное восприятие критики и так далее).

Основной упор при обучении студентов ИТ-специальностей зачастую делается на первый вид компетенций – изучаются различные языки программирования, технологии, фреймворки, интегрированные среды разработки и тому подобное. Безусловно, все эти навыки настоящему профессионалу необходимы.

Однако сами по себе они могут дать хорошего программиста-кодера, но еще не человека, готового играть ключевые роли в крупных и серьезных проектах. К тому же многие студенты практикуются на учебных задачах, не всегда дающих полное представление о промышленной разработке. Для получения более полных знаний необходимо участие в реальных проектах.

Когнитивные и социальные навыки имеют гораздо меньшее отношение к непосредственному созданию кода. Более того, такие компетенции актуальны и для многих профессий, далеких от программирования, но подразумевающих коллективную работу (хотя при этом всегда имеются нюансы, специфические для конкретной области деятельности).

Полноценное обучение подобным навыкам в рамках лекций, семинаров и практических занятий в университете вряд ли возможно. Ведь обычно они приобретаются с опытом, а опыт выполнения реальных коллективных проектов дает, как правило, гораздо больше, чем решение учебных задач (даже если последние подразумевают совместную работу нескольких студентов).

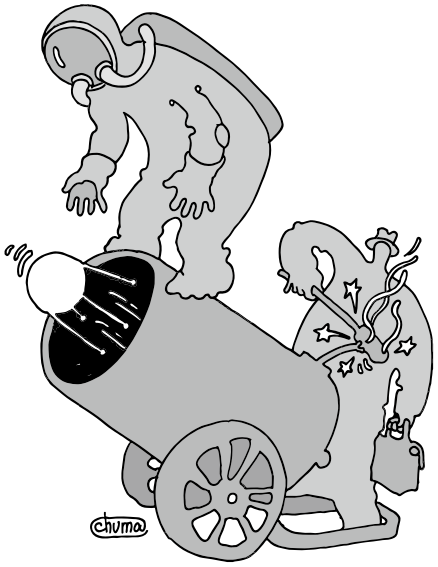
Итак, чтобы стать профессионалом в сфере ИТ, необходимо участвовать в выполнении реальных проектов – одних учебных курсов недостаточно. Как мы отмечали в предыдущей статье [1], наличие подобных проектов в портфолио студента – это существенный плюс при приеме на работу. Но где же найти такие проекты, к выполнению которых допустят людей практически без опыта? На самом деле таковых – сотни тысяч. И имя им – мир свободного и открытого ПО.

Свободные проекты

На всякий случай напомню, что свободное ПО имеет четыре основных особенности:

- свободу запускать программу в любых целях;
- свободу изучения работы программы и адаптации ее к вашим нуждам;
- свободу распространять копии, так что вы можете помочь вашему товарищу;
- свободу улучшать программу и публиковать ваши улучшения, так что все общество выиграет от этого.

Вообще в мире существуют сотни различных лицензий на программное обеспечение, считающихся свободными. Различия между ними обычно затрагивают вопросы распространения кода, основанного на коде оригинального проекта. Если вы собираетесь просто участвовать в раз-



Для получения более полных знаний необходимо участие в реальных проектах

витии уже существующего проекта, а не создавать его клон (и тем более не продавать ПО, сделанное на его основе), то с выбором лицензии можно особо не мучиться – полноценный доступ к коду и инфраструктуре разработки предоставляют практически все свободные программы.

Именно сущностью свободного ПО обусловлен тот факт, что в его развитии могут поучаствовать все желающие, ведь здесь нет опасений, что кто-то «со стороны» украдет идею или код. Безусловно, у каждого проекта имеются свои правила, и не факт, что любому желающему разрешат вносить изменения в кодовую базу. Однако никто не помещает изучать код проекта, а при желании сделать собственную копию и модифицировать ее по своему усмотрению.

В дополнение к факту свободы приложения важно обращать внимание на открытость процесса разработки, под которой подразумевается публичная доступность не просто кода, но и процессов работы над ним. Обычно это означает наличие публичной системы отслеживания ошибок, открытых почтовых рассылок и чатов, где происходит общение разработчиков, тестовых наборов и так далее.

Как следствие, можно не просто просматривать код программы, но и задавать вопросы разработчикам, а также анализировать процессы принятия решений в проекте на основе архивов почтовых рассылок и чатов. Открытость процесса разработки присуща большинству свободных проектов, хотя бывают и исключения.

Где же можно посмотреть на существующие проекты и выбрать подходящий? Мир свободного ПО очень богат и предоставляет студентам и преподавателям огромный выбор программ различной трудности и направленности.

Одним из старейших и наиболее известных хостингов для открытых разработок является SourceForge (http://sourceforge.net). В настоящее время здесь насчитывается более 300 тысяч проектов (правда, многие из них являются неактивными).

Также в последние годы большую популярность набрал GitHub (http://github.com), насчитывающий сейчас уже более двух с половиной миллионов репозиториев, а также его аналоги наподобие BitBucket (http://bitbucket.org).

Для оценки проектов с точки зрения их зрелости, сложности и активности разработки хорошо подходит сайт http://ohloh.net. Этот сервис индексирует системы контроля версий более полумиллиона проектов с общим числом строк кода, превосходящим 16 миллиардов, над которыми работают более полутора миллионов программистов.

С помощью сервиса Ohloh для каждого проекта можно посмотреть:

- основные языки программирования, используемые в разработке;
- количество активных контрибьюторов;
- активность разработки (среднее число коммитов в систему контроля версий за последнее время).

Отдельно хотелось бы отметить, что участие в открытой разработке совсем не обязательно подразумевает программирование под Linux или другую свободную ОС. Для Windows также существует немало открытых проектов – многие популярные в Linux приложения имеют версии и для этой проприетарной системы.

Это и крупные проекты (Firefox, Thunderbird, LibreOffice или GIMP), и многие небольшие приложения. Из свободного ПО, созданного специально для Windows, можно упомянуть, например, IM-клиент Miranda, файловый менеджер Far или, если вас интересуют современные управляемые среды, графический редактор Paint.NET.

Наконец, существует немало программ, написанных с использованием кроссплатформенных языков и технологий (Java, .NET, Perl/Python/PHP и так далее). Разработку таких проектов можно без проблем вести в любой удобной системе.

Познакомившись вкратце с миром свободных программ, давайте разберемся, как участие в открытой разработке поможет студентам приобрести описанные выше компетенции, необходимые для любого ИТ-профессионала.

Технические компетенции

Начнем с того, что привлечение студентов к участию в реальных проектах частично снимает с преподавателя головную боль по придумыванию задач. В любом активно

развивающемся проекте идей обычно больше, чем разработчиков, остается только выбрать проект и конкретную задачу для реализации (впрочем, такой выбор среди тысяч проектов тоже может быть непрост, но ведь не обязательно просматривать все возможные варианты).

В любом активно развивающемся проекте идей обычно больше, чем разработчиков, остается только выбрать проект и конкретную задачу для реализации

Помимо непосредственно идей, к услугам желающих их реализовать предоставляется развернутая инфраструктура для работы, включающая как минимум следующие компоненты:

- систему контроля версий – CKB (CVS, Subversion, Git и другие);
- систему учета ошибок и пожеланий (например, Bugzilla или Mantis);
- средства общения разработчиков (форумы, почтовые рассылки, чаты и так далее).

Дополнительно могут предоставляться специализированные среды сборки (Open Build Service, ROSA ABF), средства непрерывной интеграции – CI (популярными инструментами здесь являются Hudson и Jenkins) [2] и прочие специализированные инструменты, знакомство с которыми будет заметным плюсом в резюме.

Вполне естественно, что при разработке открытых продуктов используются открытые инструментальные средства (системы контроля версий и учета ошибок, компиляторы, IDE и прочее).

Во многих областях разработки открытые решения (в частности, уже упоминавшиеся CKB Git и Subversion и инструменты CI Hudson и Jenkins) удерживают лидирующие позиции, и навыки их использования с большой вероятностью пригодятся на любой работе.

Безусловно, есть сферы ИТ, в которых преобладают проприетарные решения. Участие в открытых проектах вряд ли позволит получить опыт работы с ними, но и работа с открытыми аналогами может серьезно помочь в будущем. Например, вряд ли вы найдете много открытых проектов, связанных с СУБД Oracle. Но знание открытых продуктов, наподобие MySQL и PostgreSQL, будет серьезным подспорьем в изучении этой проприетарной СУБД.

Помимо самих инструментов, многие проекты декларируют и правила работы с ними – например, составления отчетов об ошибках и реакции на них, расстановки тэгов в системе контроля версий, оформления кода и так далее. Подобные аспекты являются важной частью культуры программирования, которой многие начинающие разработчики пренебрегают.

Привить такую культуру в рамках учебных занятий сложно, а вот для участия в реальном проекте студентам так или иначе придется следовать существующим правилам, и достаточно быстро придет понимание их необходимости и обоснованности.

Когнитивные компетенции

Фактором, позволяющим развить как технические, так и когнитивные навыки, является открытость кода и сопровождающих материалов. Код, его документация, спецификации архитектуры могут служить хорошим наглядным пособием по созданию сложных продуктов. У проектов с долгой историей наверняка есть оригинальные примеры решения тех или иных вопросов – как небольших технических нюансов, так и комплексных архитектурных проблем.

Не следует забывать, что открытая разработка подразумевает не просто публичную доступность кода, но и открытость самого процесса создания ПО. В частности, в свободном доступе обычно находятся архивы почтовых рассылок разработчиков, системы отслеживания ошибок, системы контроля версий с полной историей модификаций исходного кода, документация и многие другие артефакты.

Добавьте к этому различные специализированные сайты, форумы, на которых пользователи совместно с разработчиками решают возникающие проблемы, и другие онлайн-ресурсы и вы получите огромную базу знаний, охватывающую все аспекты программной инженерии.

Эта база постоянно меняется и по своей природе имеет распределенный характер. Но поскольку она доступна через Интернет, то все ее материалы индексируются поисковыми машинами и при необходимости легко находятся. И если еще лет десять – пятнадцать назад за решением многих вопросов разработчики (по крайней мере под Windows) первым делом обращались к библиотеке MSDN, распространявшейся на нескольких дисках, то теперь гораздо проще набрать необходимый запрос в поисковой машине. Благо тот же MSDN сейчас доступен онлайн и тоже является частью всемирной базы знаний.

Справедливости ради стоит отметить, что код и структура некоторых открытых проектов могут служить примером того, как не надо писать программы; однако и изучение неудачных практик тоже очень полезно, главное – понимать, почему они неудачны.

Помимо ретроспективного анализа, полезно и наблюдение за текущей жизнью проекта. Одно дело – читать архивную переписку про выбор того или иного решения и совсем другое – наблюдать такое общение вживую и даже иметь возможность принять в нем участие, внося свои предложения и получая о них аргументированное мнение.

В заключение хотелось бы отметить, что при наблюдении за разработкой и проведением ретроспективного анализа полезно сопоставлять декларируемые планы и их реальное воплощение. Не секрет, что число ИТ-проектов, заканчивающихся полной или частичной неудачей, достаточно велико. Для того чтобы избежать таких провалов, полезно анализировать опыт предшественников, искать причины, приведшие к неудаче, а также пути их решения и предотвращения.

Для коммерческих проектов провести такой анализ вряд ли получится, даже если разработчики признают,

что проект завершился провалом либо результат имеет серьезные недочеты. А вот при открытой разработке скрыть что-то вряд ли получится – как говорится, «все ходы записаны».

С другой стороны, во многих открытых проектах часто нет четкого планирования, ведь многие разработчики трудятся над ними в свободное от основной работы время, и количество часов, отводимых на проект, может серьезно меняться в зависимости от внешних факторов. Как следствие, отставание от желаемых сроков обычно связано с недостатком времени.

Однако в крупных проектах (особенно поддерживаемых коммерческими компаниями или некоммерческими фондами) планирование разработки достаточно формализовано, и анализ процесса может помочь выявить менее тривиальные проблемы, как то: организационные промахи, неверные оценки сложности задач и количества ресурсов, или архитектурные недочеты.

Социальные компетенции

Коллективная работа. Сейчас во многих вузах активно практикуют так называемые командные проекты, когда над одной задачей совместно трудятся группы по несколько человек. Однако при этом студенты, в лучшем случае, только в теории знают, как организовать совместную работу, и на практике набивают немало шишек.

С одной стороны, это полезно с точки зрения обучения, с другой, в результате можно научиться чему-нибудь не тому. Ведь преподаватели, как правило, проверяют конечный результат командного проекта (то есть разработанную совместно программу). А вот оценить, насколько правильно и эффективно при этом был построен процесс разработки, – нелегкая задача, которая обычно затрагивается лишь поверхностно.

Присоединяясь же к реальному проекту, студенты вливаются в уже сформировавшуюся команду со своими правилами организации хода разработки. При этом можно не просто привыкать к существующим правилам, но и уточнять у опытных участников, почему выбраны именно такие подходы.

Подобный ретроспективный анализ может быть полезен и в отношении непосредственно кода проекта. Ведь нередко при изучении уже написанных программ возникают вопросы, почему было принято то или иное архитектурное решение, чем обосновано использование определенной технологии и так далее.

Как правило, новым участникам доступны как «живые свидетели» принятия решений, которых можно порасспрашивать, так и различные материалы для ретроспективного анализа, например, переписка в почтовой рассылке или обсуждение в системе учета ошибок.

Изучение таких материалов позволяет понять логику процесса разработки и предпочтения его ключевых участников. Эти знания, в свою очередь, позволяют выбирать и предлагать решения, наиболее подходящие для данного проекта, выстраивать свою аргументацию при их обсуждении и вообще эффективно взаимодействовать с остальным коллективом разработчиков, не тратя время на пространные дискуссии, не выливающиеся в создание чего-то полезного.

Английский язык. Как ни крути, а языком международного общения в мире ИТ является английский. Существенная часть документации (особенно предназначенной для разработчиков) доступна только на английском языке, а многие переводы не успевают обновляться при изменении оригиналов. Неудивительно, что и непосредственное общение в международных проектах (в форумах, рассылках, чатах и так далее) ведется на английском языке, даже если «коренных» англоязычных разработчиков немного.

Для коммерческих проектов провести ретроспективный анализ вряд ли получится, даже если разработчики признают, что проект провалился либо имеет серьезные недочеты

Например, POCA (основным продуктом которой является семейство одноименных дистрибутивов) – российская компания, и для существенной части разработчиков родным языком является русский. Однако немало и иностранных участников, не говоря уже о пользователях со всех концов света. Поэтому в рассылке для разработчиков общение ведется исключительно на английском языке, и этот же язык настоятельно рекомендуется использовать при заведении сообщений об ошибках.

Конечно, применительно к пользователям жестких ограничений нет – они могут общаться на родном языке в форумах, а вместо непосредственного заведения сообщения об ошибке обратиться в службу технической поддержки [3]. Но если пользователь хочет, чтобы к решению его проблемы подключилось как можно больше людей, то лучше все-таки использовать английский.

Не секрет, что реальная практика является отличным подспорьем в изучении иностранного языка. Даже просто ежедневно читая сообщения других участников проекта (не говоря уже о разговорном общении), вы постепенно совершенствуете свои познания. Конечно, лучше обучаться, контактируя с людьми, хорошо знающими язык, иначе можно научиться такому, что даже англичане и американцы вас вряд ли поймут. Впрочем, в подавляющем большинстве проектов с этим проблем нет.

Программы вовлечения студентов

К сожалению, далеко не все преподаватели активно привлекают студентов к участию в реальных проектах. Однако что мешает проявить личную инициативу?

Можно просто присоединиться к какой-нибудь команде ради интереса и приобретения опыта (но и с прицелом на то, что в случае успеха вы получите хороший плюс в резюме). Можно попробовать убедить преподавателя зачесть участие в понравившемся вам открытом проекте как работу, выполненную в рамках его предмета. Наконец, можно не ограничи-

ваться далекими перспективами и учебно-познавательными целями и воплотить свои инициативы во вполне ощутимую финансовую выгоду. Сделать это можно посредством участия в различных программах, в рамках которых оплачивается работа студентов над свободными приложениями.

Участие в свободном проекте — это плюс в резюме студента, подтверждающий как владение передовыми технологиями, так и наличие навыков работы в распределенной международной команде

Наиболее известной программой такого рода является Google Summer of Code (GSoC), с 2005 года проводимая интернет-гигантом. В рамках этого мероприятия оплачивается летняя работа студентов и аспирантов над разнообразными открытыми проектами. Размер гранта, получаемого студентом в случае успешного выполнения работ, – 5000 долларов, что является неплохой суммой для представителей нашей страны. Подразумевается, что студенты работают летом пять дней в неделю по восемь часов в день; однако работа происходит удаленно, и как в действительности будет распределено рабочее время – это дело участника. Главное, чтобы менторы, которые принимают работу, были довольны результатами.

Хотя собственно работа происходит летом, отбор проектов для нее стартует еще в феврале, а где-то в середине весны начинается прием заявок от студентов. Студентам, желающим участвовать в программе, рекомендуется подбирать себе проект заранее – в частности, полезно предварительно связаться с разработчиками интересных вам проектов, обсудить детальные задачи и возможность вашего участия. Самая удобная для обеих сторон ситуация, когда потенциальный участник GSoC и так работает над проектом в свободное время и уже хорошо знаком как с кодом, так и с разработчиками.

В дополнение к GSoC некоторые крупные проекты проводят собственные летние программы, в рамках которых студентам предлагается поработать над улучшением конкретных приложений. К регулярным мероприятиям такого рода относятся, например, Season of KDE и Ruby Summer of Code. Интересно, что в последние годы появились программы, нацеленные на вовлечение в мир свободного ПО представительниц прекрасного пола – например, RailsGirls Summer of Code или Outreach Program for Women.

Помимо масштабных программ, подразумевающих полную занятость студентов в течение достаточно долгого времени, бывают и эпизодические работы. Например, многие проекты платят премии за обнаружение в их продуктах серьезных ошибок, в первую очередь связанных с проблемами безопасности. Такие инициативы распространены как в открытых проектах (в частности, в фонде Mozilla), так и в крупных корпорациях, производящих закрытое ПО, – к их числу относятся Microsoft, Facebook, Paypal и многие другие.

В некоторых больших проектах (например, в Ubuntu) предусмотрено вознаграждение от пользователей за реализацию той или иной функциональности – то есть пользователи открыто заявляют, что готовы заплатить определенную сумму за указанную работу. Однако обычно такие запросы относятся к нетривиальным вещам, реализация которых может потребовать специфических знаний. Прежде чем браться за такие задания, следует хорошо ознакомиться с кодом проекта и оценить трудоемкость и сложность реализации; вряд ли разумно хвататься за подобные задачи просто из желания заработать немного денег.

***

Помимо получения различного рода компетенций, участие в открытых проектах имеет и ряд других преимуществ перед выполнением учебных заданий либо стажировкой в закрытых компаниях.

По сравнению с учебными заданиями у студентов есть возможность поучаствовать в создании реально востребованных приложений с большой пользовательской аудиторией. Этот фактор нередко заставляет более ответственно относиться к работе, ведь качественный результат, с одобрением принятый пользователями, даст немалое моральное удовлетворение от проделанной работы, а вот неудачное решение может быть подвергнуто критике (возможно, что достаточно жесткой) и еще долго будет напоминать о себе.

Участие в свободном проекте – это огромный плюс в резюме студента, подтверждающий как владение передовыми технологиями программирования, так и наличие навыков работы в распределенной международной команде. При этом результаты работы (в частности, программный код) доступны публично и являются наглядной демонстрацией умений. При найме на работу программистов именно изучение выполненных проектов является одним из ключевых факторов, зачастую перевешивающим наличие всевозможных сертификатов и дипломов. Думаю, здесь вполне уместно вспомнить одно из высказываний Линуса Торвальдса: «Talk is cheap. Show me the code» («Слова ничего не стоят. Покажите мне код»).

Наконец, внося результативный вклад в открытый проект, вы фактически участвуете в развитии «технологической копилки» всего человечества и помогаете как людям, использующим вашу программу, так и разработчикам, анализирующим, повторно использующим и улучшающим ваши решения. Надеюсь, что для многих студентов этот фактор – ничуть не меньший повод для участия в открытой разработке, чем плюс в резюме. EOF

1. Силаков Д. Корпорации с человеческим лицом. Секреты успешного собеседования. //«Системный администратор», №7-8, 2013 г. – С.135-139.
2. Ушаков М. Continuous Jntegration системы Hudson. Развертывание и настройка. //«Системный администратор», №4, 2013 г. – С. 87-91.
3. Силаков Д. Контроль качества дистрибутива Linux. //«Системный администратор», №4, 2013 г. – С. 82-86.