Ссылки и примечания

[1] Движение луддитов зародилось в Англии как форма протеста работников против механизации текстильной промышленности, в частности, против внедрения жаккардовых станков. Протесты вылились в массовые бунты, в ходе которых работники разрушали станки, угрожавшие их благосостоянию. Некоторые группы луддитов начали разрушать также железные дороги и любые станки, приходившие на смену малоквалифицированному труду. Руководителем движения считался, очевидно, мифический лидер по имени Нед Лудд, известный также как Генерал Лудд или Король Лудд, который, по слухам, подобно Робин Гуду скрывался в Шервудском лесу. Движение луддитов было разгромлено в 1816 г., когда по решению английского парламента на подавление бунтов были брошены регулярные воинские части численностью около 12 тыс. человек. Лидеры движения были казнены или сосланы в Австралию. С тех пор луддитами стали называть всех, кто выступает против применения новых технологий.

[2] Toffler A. Power Shift: Knowledge, Wealth, and Violence at the End of the 21st Century. New York: Bantam Books, 1990.

[3] Drucker P.F. The Information Executives Truly Need // Harvard Business Review. January-February, 1995.

[4] Gates B. Managing @ the Speed of Thought. New York: Warner Books, 1999.

[5] Naisbitt J. Megatrends. New York: Warner Books, 1982.

[6] Negroponte N. Being Digital. New York: Random House, 1995.

[7] Ibid, 94.

[8] Pekka A. Software Quality through Insight, Foresight, and Leadership // EXBA — Quality Connection Special Issue. May 2002.

[9] Gates B. Managing. 102.

[10] Garvin D. A. Competing on the Eight Dimensions of Quality // Harvard Business Review. November-December, 1987.

[11] Устранение посредников позволяет пользователю Интернета или покупателю получать прямой доступ к информации, для получения которой ранее требовалось промежуточное звено (медиатор), в роли которого мог выступать, например, продавец, библиотекарь или юрист. Современные технологии дают возможность пользователю напрямую просматривать интересующую его информацию, а также сравнивать товары разных производителей, не нуждаясь в услугах медиатора (врача, юриста, продавца). По крайней мере Интернет принципиально меняет характер взаимоотношений между потребителями и поставщиками товаров и услуг по сравнению с традиционными подходами, существовавшими до этого.

[12] Garvin D. Eight Dimensions. [13]

Ibid, 227.

[14] Tappscott D. The Digital Economy. New York: McGraw-Hill, 1996.

[15] Ibid, 272-73.

[16] Ibid, 279.

Глава 9

Обеспечение качества программного продукта на примере создания системы Linux

Роберт И. Коул,
Гвендолин К. Ли

Введение

Проект создания ядра операционной системы Linux представляет одну из первых масштабных попыток глобального объединения талантов и творческих способностей многочисленных программистов для совместной разработки важнейшего открытого программного продукта (OSS)[1]. В ходе выполнения проекта тысячи талантливых добровольцев сумели преодолеть корпоративные и географические границы, чтобы объединить при посредстве Интернета свои усилия и создать наукоемкий, новаторский, высококачественный программный продукт (ПП). Такой подход, принципиальным отличиям которого от традиционных способов создания коммерческих программных средств посвящена настоящая глава, существенно и положительным образом повлиял на качество конечного продукта.

Система Linux представляет серьезную угрозу доминирующему положению UNIX и MS Windows NT на рынке операционных систем, поскольку ее поддерживают и продвигают такие серьезные фирмы, как Sun Microsystems, Oracle, Hewlett-Packard и IBM. Примечательно, что система Linux, создание которой в 1991 г. начиналось как хобби разработчиков, к 1999 г. в качестве открытого ПП сумела стать ведущей операционной системой Всемирной паутины. Эту систему использует 31% веб-сайтов, в то время как под Windows создано 24%, а под Solaris — 17% сайтов[2]. Более того, Linux продолжает завоевывать новые рынки. В частности, она особенно популярна в научных и университетских кругах, а как доказывает история, успешное применение ПП в этой среде становит-

ся залогом его последующего широкого распространения в бизнесе. В этом смысле достаточно напомнить пример с распространением операционной системы UNIX.

С учетом современных масштабов и перспектив дальнейшего расширения областей применения Linux, можно констатировать успешность этой системы. Распространенность системы сама по себе может служить косвенным подтверждением ее высокого качества, но имеются данные, которые непосредственно подтверждают высказанное предположение. По данным опросов, профессионалы, работающие в области деловых технологий, отмечают высокую надежность системы — второй по значимости фактор, побуждающий многие компании применять Linux, после относительно низкой стоимости и отсутствия платы за лицензии на ее использование[3]. Третий по значимости фактор — высокие эксплуатационные характеристики системы. Исследование, проведенное Министерством обороны США, доказало, что большие возможности и высокая надежность являются основными аргументами в пользу ПП с открытыми исходными кодами, к числу которых принадлежит Linux[4]. Значение, которое придают пользователи надежности, становится понятным, если учесть, что специалисты по деловым технологиям, отвечая на вопрос, в каких, коммерческих или разрабатываемых на заказ, ПП чаще всего приходится сталкиваться с дефектами и ошибками, в первую очередь называют именно операционные системы[5]. Более того, имеются достаточно убедительные свидетельства, доказывающие, что подобные ошибки в операционных системах влекут за собой значительные убытки для компаний.

Нельзя категорически утверждать, что система Linux полностью свободна от проблем, связанных с качеством. При опросах специалисты отмечали малое число разработанных под Linux прикладных программ с теми свойствами, которые нужны для использования в бизнесе, ограниченные возможности для обучения пользователей, недостаточность технической поддержки системы[6]. Linux нашла свою рыночную нишу преимущественно в качестве операционных систем для веб-серверов, обслуживания внешних интерфейсов, обмена файлами и их распечатки[7]. По мере создания приложений, написанных под Linux, эта операционная система начинает шире применяться в качестве клиентских операционных систем и СУБД[8]. В целом же качество операционной системы следует рассматривать как одно из главных условий ее широкого распространения.

Авторы настоящей главы предприняли попытку проанализировать процесс обеспечения высокого качества OSS на примере опыта создания ядра операционной системы Linux. Чтобы выявить принципиальные отличия модели разработки OSS, авторы сопоставили ее с обычными схемами создания коммерческих ПП и в более широком плане — с традиционными моделями проведения НИОКР, намереваясь определить условия, благодаря которым удается обеспечить высокое качество разработки OSS.

Процесс создания OSS поражает воображение программистов по всему миру, независимо от места их работы, политических убеждений, экономического

статуса или национальности. Ни одна фирма в мире не способна достичь подобных масштабов сотрудничества высококлассных специалистов. Любая компания всегда ограничена в подборе исполнителей, поскольку вынуждена привлекать специалистов из числа имеющихся на местном рынке трудовых ресурсов или в лучшем случае из компаний, входящих в цепочку поставок ее продукции, либо потребителей. Одной из главных черт процесса создания OSS, принципиально отличающих его от традиционных моделей разработки коммерческих ПП и любых других НИОКР, является то, что пользователи продукта выступают одновременно и в роли его создателей.

От прочих ПП OSS отличает доступность исходных кодов для пользователей. Когда ПП распространяется в исходных, а не в машинных кодах, каждый пользователь имеет возможность модифицировать или расширить его в соответствии с собственными потребностями. В отличие от OSS, большинство коммерческих ПП распространяется, как правило, в машинных кодах, доступных для понимания и исполнения только компьютеру. С точки зрения производителей коммерческих ПП, такой подход служит средством защиты интеллектуальной собственности, поскольку машинные коды имеют двоичную форму, и редкий пользователь способен восстановить исходные коды программы. Однако при этом сторонние программисты лишаются права повысить надежность продукта, придать ему дополнительные возможности или обновить его функциональные свойства, в то время как главными тремя признаками высококачественного продукта следует считать именно его эффективность, эксплуатационные свойства и надежность[9].

Анализ опыта создания ядра Linux показал, что участие пользователей OSS в их разработке сводится к двум важнейшим функциям: 1) обеспечению качества и 2) инновациям. Применительно к обеспечению качества задачи, решаемые пользователями, заключаются в передаче сообщений об ошибках ПП, участии в их выявлении и устранении и последующих проверках. Инновационная деятельность пользователей состоит в подаче предложений о придании продукту новых свойств и в написании так называемых «заплат», позволяющих улучшить его эксплуатационные качества. Обе перечисленные задачи одинаково важны для обеспечения конкурентоспособного уровня качества ПП. В отличие от традиционной разработки ПП специализированными фирмами, которые считают пользователей только источниками рекламаций или сообщений об ошибках, модель создания OSS предусматривает непосредственное участие пользователей в решении возникающих проблем. Они также служат источниками предложений по доработкам и инновациям.

Трудно переоценить преимущества, которые с точки зрения обеспечения качества ПП можно получить, когда пользователи становятся соавторами разработки[10]. В случае применения обычных коммерческих ПП их неопытные пользователи часто не могут выявить имеющиеся дефекты или обнаруживают их слишком поздно. Когда же им все-таки удается найти ошибку в программе, их сообщения

о ней оказываются нечеткими и нуждаются в переводе на язык, понятный разработчикам. Информация об ошибках часто проходит через много промежуточных инстанций, пока не поступит к специалистам, которые обязаны диагностировать возникшую проблему на основании отфильтрованных данных, попытаться установить причину дефекта и переправить всю информацию лицу, ответственному за принятие решений. В таком протяженном по времени процессе неизбежны искажения информации, задержки с принятием решений или отсутствие реальных действий. Вовлечение неопытных пользователей в процесс выявления проблем и наличие большого числа посредников, изучающих поступившую информацию об этих проблемах, в равной мере повышают вероятность запаздывания решений, непринятия вообще или неудачных решений. Нет ничего удивительного в том, что в качестве наиболее распространенных причин отказов в соответствующих базах данных фирм — разработчиков ПП указывается «дефект не обнаружен (NFF)» или «дефект не может быть воспроизведен (CND)». Зачастую число дефектов именно этих двух категорий превышает 30% от общего числа сообщений об отказах, содержащихся в базах данных. Часто также в качестве причин дефектов указывают отказы отдельных технических устройств — клавиатуры, монитора, системной платы, памяти и т.д.[11]. Такое положение дел с расследованием ошибок ПП отражает отсутствие взаимодействия между теми, кто испытывает на себе последствия имеющихся проблем, и теми, кто пытается их разрешить. Модель разработки Linux, в которой пользователи являются также соавторами создания системы, устраняет подобное отсутствие взаимодействия между заинтересованными сторонами. В результате удается свести к минимуму дефекты ПП, относящиеся к категориям NFF и CND. Кроме того, при обычном подходе к разработке ПП для решения возникающих проблем привлекается вся иерархическая структура управления компанией, причем менеджеры, стоящие на верхних ступенях иерархии, руководствуются при выработке решений совсем иными критериями. Например, их больше волнуют стоимость решения проблем или необходимость отвлечения специалистов, занятых другими работами.

Квалифицированных пользователей и одновременно соавторов OSS отличает способность скорее и точнее выявлять проблемы и доводить полученную информацию более быстро и четко до своих коллег в том виде, который способствует коллективному исследованию проблемы и своевременному принятию предупреждающих действий. Например, от всех пользователей и соразработчиков Linux требуется наряду с сообщением о выявленной ошибке подавать свои предложения о мерах по ее устранению. Коллективное решение проблем специалистами одного уровня с большей вероятностью происходит на основе фактических данных, а сам поиск решения ведется намного быстрее, чем в рамках иерархической структуры управления компаниями. Модель разработки OSS позволяет минимизировать источник искажений при выработке решений, чего нельзя сказать о подходах к рассмотрению и устранению ошибок, применяемых фирмами — разработчиками коммерческих ПП.

Проекты создания OSS нельзя считать единственными, в которых имеют место инновации, инициируемые пользователями. Однако эти проекты представляют собой крайние случаи феномена участия пользователей в разработках, позволяющие устранить любые ограничения на размеры добавленной ценности, которую вносят пользователи в создаваемый совместными усилиями продукт. Такой подход полностью согласуется с самыми современными определениями понятия качества как меры удовлетворенности потребителей[12]. Опора на пользователей при создании OSS является главной особенностью этого процесса, поскольку при этом конечный продукт приобретает полезные свойства, в которых заинтересованы пользователи, а это ведет, в свою очередь, к повышению уровня их удовлетворенности.

Для того чтобы выявить организационные условия, содействующие созданию продуктов с устойчиво высоким уровнем качества, следует получить ответы на следующие три вопроса: 1) Кто принимает участие в создании OSS и что заставляет их добровольно участвовать в этой работе? 2) Как должны быть организованы и скоординированы усилия добровольных участников процесса разработки? 3) Как разработчикам удается обеспечить высокое качество процесса разработки?

Поиск ответа на первый из поставленных вопросов должен выявить способы мобилизации усилий квалифицированных программистов, разделенных корпоративными и географическими границами, и формирования из них широкомасштабного сообщества разработчиков OSS. Это сообщество не является объединением случайных людей. Из более 5 млн программистов, насчитывающихся во всех странах мира, менее 50 тыс. принимают участие в проектах создания открытых систем[13]. Что служит мотивом, побуждающим географически и организационно рассеянных по планете специалистов объединить свои усилия таким образом, чтобы их результатом стало создание столь высококачественного продукта, как Linux?

Второй вопрос привлекает внимание к изучению организационных методов и структур, используемых для координации усилий разработчиков OSS. Мун (Moon) и Спроулл (Sproull) объясняют успешность создания Linux присутствием сильных лидеров[14], а Маркусу (Marcus) с соавторами удалось выявить типичную модель управления, применявшуюся при создании нескольких OSS[15]. Как лидерство, так и применение определенных моделей управления представляют собой схему принятия решений «сверху». Модели повышения качества, основанные на принятии решений сверху, удается быстрее внедрить, но их использование ограничено сопротивлением работников нижестоящих уровней, поскольку навязываемые сверху решения им чужды. Они часто подвергают сомнению обоснованность директив, спускаемых вышестоящими руководителями[16]. Более того, рассмотрение

процесса создания Linux как управляемого сверху не полностью отражает роли главных участников феномена OSS — пользователей, действия которых являются залогом успешного внедрения инноваций в разработанную систему. Далее будут рассмотрены методы и схемы, применяемые для координации работ по созданию системы, причем особое внимание будет уделено тем из них, благодаря которым инновационный процесс позволяет вовлечь в работу большее число участников, чьи интересы и уровни квалификации могут серьезно различаться.

Третий из поднятых вопросов относится к тому, каким образом достигается высокое качество продукта, получаемого на выходе указанного инновационного процесса. По мнению Реймонда (Raymond), Linux превосходит многие операционные системы по уровню качества и другим характеристикам благодаря открытости и эволюционному характеру процесса разработки[17, 18]. Большинство коммерческих и некоммерческих ПП разрабатывается, по терминологии Реймонда, командами программистов, работающих над текстами программ изолированно, постепенно доводя продукт до окончательной версии. Напротив, разработка Linux велась, используя терминологию Реймонда, «базарным» способом, при котором часто реализовывались многочисленные промежуточные версии, циклы разработки которых были довольно непродолжительными по времени. Кроме того, традиционные модели коммерческих ПП остаются преимущественно ориентированными на создание специализированными фирмами, которые, тем не менее, уже начинают использовать некоторые подходы, апробированные в ходе работ над Linux. В частности, фирмы — разработчики ПП стали шире использовать возможности Интернета, создавая многосторонние цифровые платформы для наукоемких продуктов, привлекать к разработке приложений специалистов многих других фирм, чтобы использовать географически распределенные знания в области программирования. Они также все шире привлекают будущих основных пользователей разрабатываемых продуктов к их оценке и выявлению ошибок. Например, в процессе разработки операционной системы Windows 2000 Microsoft объявила об учреждении совместной программы, охватывающей 50 крупнейших потребителей ПП корпорации, которые предлагали свои идеи в отношении устранения недостатков системы и придания ей новых возможностей[19]. Но, безусловно, эта цифра несравнима с числом участников создания Linux.

Соглашаясь с Реймондом, следует рассматривать создание ядра Linux как эволюционный процесс, в ходе которого происходит непрерывное и постепенное совершенствование конечного продукта. Этот процесс резко контрастирует с довольно слабыми инициативами фирм — разработчиков коммерческих ПП по созданию высоконадежных систем. Подобные фирмы постоянно испытывают давление конкурентов, заставляющее их как можно скорее выходить на рынок с новыми продуктами, зачастую пренебрегая контролем их качества в процессе разработки. Более того, применяемая при создании многих из имеющихся на рынке ПП модель скорейшей окупаемости инвестиций находится в постоянном противоречии с требованиями обеспечения их высокой надежности[20].

Расширяя рассмотрение эволюционной схемы создания OSS типа Linux, авторы постараются проверить свои выводы, касающиеся организации и структуры разработок таких систем в контексте того, как возникают предложения по изменению исходных кодов, как отбираются и сохраняются из этих предложений лучшие, с тем чтобы создать ПП, сочетающий высокую надежность с инновационным характером.

Обратимся теперь к конкретному примеру проекта разработки ядра операционной системы Linux и постараемся в ходе его рассмотрения предложить ответы на три основных вопроса, поставленных выше.

Пример проекта разработки ядра операционной системы Linux

Рассмотрим проект создания ядра операционной системы Linux (далее — ядра Linux), обращая внимание на следующие основные проблемы:

1) способы мобилизации большого числа пользователей, обладающих необходимой квалификацией для участия в разработке данного ПП;
2) методы организации работ и структура, обеспечивающие координацию усилий участников проекта, предлагающих различные инновации, и тех, кто занимается отбором и сохранением предложенных новшеств;
3) применение открытого эволюционного процесса разработки ПП.

Для изучения проекта создания ядра Linux авторы пользовались большим числом источников информации, главным из которых послужили артефакты, созданные непосредственными исполнителями проекта. Эти артефакты являются основными результатами инновационной активности разработчиков, и, несомненно, самым важным из них следует считать исходные коды самой операционной системы Linux[21].

Напомним, что исходным принято называть код, который используют программисты для написания компьютерных программ. В качестве основного источника информации была выбрана версия Linux 2.2.14, реализованная в марте 2000 г., поскольку процесс создания ядра Linux стабилизировался на версии 2.2, варианты которой разрабатывались в период между 1999 и 2000 гг. Последующие наиболее существенные доработки Linux уже не затрагивали ядра системы[22]. Исходный код Linux 2.2.14 имеет объем 62,7 мегабайта и содержит 1,9 млн программных строк в 5186 файлах, распределенных по 266 папкам. Совместно с исходными кодами системы пользователям направляют текстовые файлы Credits и Maintainers. Для простоты обращения эти файлы расположены на первом уровне директории системы вместе с папками, содержащими программные модули и документацию.

Файл Credits содержит список людей, внесших значительный, признаваемый всеми вклад в создание ядра Linux[23]. В этом списке наряду с именами признанных разработчиков указано, в чем заключен существенный вклад каждого из них в разработку ядра системы. Файл Maintainers содержит описания всех основных подсистем и указывает, кто отвечает за их поддержку.

Помимо исходного кода Linux 2.2.14, полезным источником информации служит список рассылки версий ядра системы и архив электронной переписки по нему. Список рассылки был в свое время составлен для организации обсуждения проблем, возникавших в процессе разработки системы, а архив электронных почтовых сообщений позволяет оценить масштабы сообщества разработчиков, участвовавших в создании Linux, и изучить, как менялись со временем основные направления разработок. Список рассылки служил виртуальной средой, в которой разработчики системы рассылали свои предложения, детально обсуждали вопросы их внедрения и взаимодействовали между собой[24]. Были учтены все, кто в период между 1995 и 2000 гг. направил хотя бы одно электронное сообщение, касающееся разработки ядра Linux. Всего было выявлено 14 535 авторов, каждый из которых за пять лет прислал в среднем по 14 сообщений[25].

Еще одним источником данных послужили расширения электронных адресов отправителей сообщений, которые были извлечены из списка рассылки ядра Linux и файлов Credits и Maintainers. С их помощью выявлялись организационная и национальная принадлежность авторов сообщений. Кроме того, использовались данные он-лайновых опросов участников разработки ядра Linux. В частности, группа исследователей из Университета германского города Киль собрала анкетные данные участников проекта, разослав по всем адресам, содержащимся в списке рассылки ядра системы, объявление о проводимом ею исследовании и просьбу ко всем участникам разработки заполнить электронную анкету, размещенную на созданном группой веб-сайте проекта. Это исследование проводилось в период с февраля по апрель 2000 г., а его результаты позволили проанализировать демографию, должностной статус и мотивацию участников разработки ядра Linux.

Мобилизация сообщества разработчиков ядра Linux

В проекте разработки ядра Linux приняло участие несколько тысяч добровольцев, распределенных по многим организациям и странам мира. Опыт реализации проекта доказал целесообразность масштабной он-лайновой кооперации усилий многочисленных, выступающих в одном лице разработчиков и пользователей системы. К 2000 г. в мире насчитывалось более 12 млн пользователей Linux, из которых только около 90 тыс. было зарегистрировано официально[26]. Из общего числа зарегистрированных пользователей примерно 16% выступили

также в роли разработчиков системы[27]. К сожалению, со временем число пассивных пользователей растет быстрее числа активных пользователей, являющихся одновременно участниками доработок системы. В данном разделе будет рассмотрено, кто является добровольными участниками команды исполнителей проекта и какими мотивами они руководствуются, участвуя в разработке OSS.

Далее будет также проанализировано, что привлекает добровольцев в проекте и как поддерживается устойчивость их участия в его выполнении. В существующей литературе по вопросам качества указывается на то, что способы мотивации сотрудников и природа используемых при этом стимулов играют определяющие роли в обеспечении качества работы предприятий. Согласно данным Маркуса и его соавторов, источником мотивации для добровольных участников создания ядра Linux послужили стимулы социального и экономического характера[28]. Сопоставляя стимулы, определяющие мотивацию добровольных разработчиков Linux и работников конкретного предприятия, легко заметить несовпадение механизмов их действия.

Экономические и социальные выгоды относятся к числу внешних стимулов. Среди наиболее часто встречающихся ответов он-лайнового опроса участников разработки ядра Linux преобладают следующие заявления: «Участие в разработке способствует повышению моего профессионального уровня как программиста»; «Оно содействует повседневной работе»; «Дает возможность приобретения преимуществ для карьерного роста»; «Позволяет общаться с другими разработчиками программных продуктов» (табл. 9.1). Стимулами для работников коммерческих фирм в основном служат размеры заработной платы, возможность льготного приобретения акций своей компании, карьерный рост, участие в интересных проектах. В отличие от постоянных работников, добровольцев привлекает не заработная плата, а экономические преимущества от применения более совершенных ПП. Повышение качества применяемых программных средств означает для частных пользователей возможность работать лучше, а для корпоративных пользователей, чьи деловые интересы тесно связаны с применением Linux, — повышение эффективности деятельности компании.

Добровольцы нуждаются в кооперации и в обмене предлагаемыми решениями для разработки высококачественного ПП, поскольку сложность создаваемой системы такова, что требует коллективных усилий. В частности, добровольцы видят необходимость тратить собственное время и использовать имеющиеся у них профессиональные навыки для участия в создании ПП более высокого качества, поскольку им известны общие проблемы, требующие решения. Они добровольно принимают участие в коллективном решении проблем, предлагают свои варианты решений и получают на них отклики других участников проекта в надежде на избавление от подобных проблем в будущем. Итак, индивидуальные потребности участников проекта в более совершенных ПП удовлетворяются в рамках общественного процесса, в котором взаимная заинтересованность

Таблица 9.1

Мотивы и ожидания участников создания ядра Linux (по состоянию на 2000 г.)

Мотивы и ожидания	Респонденты полностью согласные с данными утверждениями, %				
	Все респонденты (154 чел.)	Студенты (36 чел.)	Не имеющие постоянного места работы (93 чел.)	Работающие временно (26 чел.)	Работающие постоянно (26 чел.)
Внутренние стимулы					
Для меня важно получать удовольствие от программирования	66,2 57,8 1,9	75,0 66,7 **0**	67,7 63,4 2.2	76,9 61,5 **0**	61,5 46,2 7,7
Я участвую в создании открытого ПП потому, что убежден в обязательности свободы информации	66,2 64,9 53,2 42,2 23,4	77,8 63,9 69,4 38,9 25,0	68,8 63,4 51,6 46,2 20,4	73,1 73,1 69,2 34,6 15,4	73,1 73,1 53,8 46,2 42,3
Отсутствие оплаты за участие в проекте Linux стало для меня серьезным разочарованием	22,1	27,8	23,7	11,5	34,6
Внешние стимулы					
Для меня очень важно повысить собственные навыки программирования					
Участие в проекте способствует моей повседневной работе, поскольку для меня очень важно располагать более совершенными программными средствами					
Повышение качества ядра Linux					
Личное общение с другими разработчиками ПП					
Преимущества для карьерного роста					
Завоевание репутации опытного программиста внутри сообщества разработчиков Linux					

Источник: http://www.psychologie.uni-kiel.de/linux-study/writeup.html

сторон способствует обмену знаниями и информацией. В этом процессе у его добровольных участников вырабатывается чувство сопричастности к сообществу единомышленников, они испытывают удовлетворение от взаимодействия с другими столь же грамотными специалистами.

Приобретаемые в ходе участия в проекте известность и репутация также сулят добровольцам вполне определенные социальные и экономические выгоды. В частности, это повышает «цену» добровольных участников проекта на рынке труда за пределами сообщества разработчиков проекта[29, 30]. Внутри же самого этого сообщества высокая репутация участника означает также, что можно говорить о нем как о классном специалисте, что способствует повышению его общественного статуса. А самое важное то, что в психологическом плане повышение репутации позволяет человеку удовлетворять его индивидуальные потребности во внимании, возможности работать в команде и признании коллег. По большому счету «социальный статус человека определяется не тем, чем он управляет, а тем, какую память он о себе оставляет»[31]. Именно это поддерживает добровольцев в их постоянном участии в проекте. В противном случае они покидают сообщество, после того как сочтут, что приобретенной ими репутации достаточно для признания на рынке труда.

Другая составляющая мотивации — внутренние стимулы. В отличие от внешних стимулов, внутренние заставляют добровольцев участвовать в проекте, поскольку для них ценен сам факт участия в столь престижной работе. К числу подобных стимулов можно отнести стремление к новизне, жажду приключений, желание удовлетворить собственную любознательность, повысить профессиональное мастерство. По мнению авторов, под эту категорию стимулов подпадают заявления опрошенных участников проекта о том, что их привлекает «удовольствие от самого процесса программирования», «убежденность в необходимости свободы информации», «отрицание того, что деньги решают все». В частности, «радость от мастерски выполненной работы проистекает из доказательства самому себе собственных талантов и способностей»[32].

Как описанная схема мотивации разработчиков способна обеспечить устойчиво высокое качество ПП, демонстрируемое Linux? Поиск ответа на этот вопрос подводит нас к следующему аспекту рассмотрения данного проекта.

Структура сообщества разработчиков ядра Linux

Рассмотрим, как осуществляется координация усилий по созданию ПП в сообществе добровольцев. В частности, следует эмпирически оценить организацию этого сообщества, позволяющую ему при постоянно расширяющемся составе участников и их разнообразии справляться с преобразованием большого числа поступающих от них предложений в высококачественный ПП. Ежегодно проводимые подсчеты числа электронных сообщений в списке рассылки ядра

Linux доказывают, что численность сообщества разработчиков системы за период с 1995 по 2000 г. увеличилась в четыре раза. Чем больше размеры сообщества, тем сложнее координировать и изучать все возможные взаимосвязи между составными частями ПП, разрабатываемыми разными представителями этого сообщества. То, что разработчики Linux сумели справиться с этой проблемой без ущерба для качества системы, можно считать наиболее впечатляющим их достижением.

Официальная структура, принятая сообществом для решения проблем координации, основана на принципе модульного построения. Принятие решений на уровне отдельных модулей децентрализовано. Модульная структура гарантирует разработчикам полную свободу одновременной работы над отдельными модулями системы без риска создания помех для работы над другими модулями. Без применения модульного принципа построения системы и при большом числе разработчиков, совместно трудящихся над созданием компьютерной программы, любые, самые мелкие изменения в одной ее части могут создавать существенные проблемы с качеством, приводя к необходимости изменять и даже полностью перерабатывать остальные части программы. Поэтому главный идеолог Linux Лайнус Торвалдс (Linus Torvalds) решил использовать загружаемые модули версии 2.0.0 Linux, реализованной в 1996 г., с тем чтобы установить границы, в пределах которых разработчики каждого модуля получили полную свободу в их отработке и внедрении. Благодаря этому модульная структура системы обеспечила надлежащую координацию между отдельными ее составными частями, позволившую минимизировать проблемы качества.

Кроме того, модульность позволяет системе непрерывно развиваться. Без этого в случаях, когда разработчик оригинального модуля покидает сообщество, а другие разработчики присоединяются к нему позднее, им было бы трудно включиться в работу из-за слишком высокого уровня сложности всей системы. Известно, что при прочих равных условиях компании с высокой текучестью кадров испытывают более серьезные трудности с поддержанием качества своих продуктов[13]. Это особенно справедливо при разработке ПП, создаваемых, подобно Linux, в виртуальной среде, когда разработчики могут свободно в любой момент покинуть сообщество или присоединиться к нему. Модульный принцип построения системы позволяет четко и понятно формулировать задачи, стоящие перед новыми членами сообщества. Таким образом, модульность служит важным структурным механизмом разработки, который позволяет минимизировать проблемы качества путем упрощения решаемых задач и предупреждения нарушения непрерывности процесса разработки в связи со сменой исполнителей.

Другой официально признанной сообществом разработчиков Linux структурой является параллельное существование двух кодовых деревьев, из которых одно предназначено для экспериментирования, а другое — для уже отработанных программ (основное устойчивое дерево). Экспериментальное кодовое дерево служит тем местом, где разработчики могут экспериментировать с новыми технологиями и проверять возникшие идеи. Процесс обучения разработчиков на собственных ошибках может быть легко реализован на экспериментальном кодовом дереве, не создавая при этом помех для пользователей основной части ПП. На экспериментальном дереве проверяют новейшие свойства ПП и инновационные решения, в то время как на основном устойчивом дереве сосредоточены более зрелые программы. Структура, принятая сообществом разработчиков ядра Linux, представляется наиболее подходящей для реализации таких задач, как поощрение вариативности решений, постепенное совершенствование системы и накопление опыта разработчиками. Все эти задачи являются важными составляющими успешных инноваций.

Помимо официальных структур, сообщество разработчиков ядра Linux опирается на неформальные, подразумеваемые схемы координации своей деятельности. Неформальность этих схем обусловлена тем, что состав исполнителей формируется непосредственно в процессе выполнения отдельных задач. Не существует официального закрепления каких-либо задач за определенными добровольными их исполнителями. Для того чтобы проследить эти неформальные отношения между разработчиками, был применен метод, основанный на просмотре заголовков электронных сообщений, присутствующих в архиве электронной переписки по Linux. При этом было выявлено, что добровольные участники разработки системы вносят свой вклад в решение самых разнообразных задач, таких как поиск и устранение ошибок, испытания системы, написание руководств, добавление новых возможностей, размещение операционной системы на различных компьютерных платформах.

Часть исполнителей специализируются на решении задач определенного типа, поскольку для них требуются специальные навыки программирования. Например, выявлять ошибки и присылать соответствующие сообщения, подавать запросы на придание системе дополнительных возможностей способно большинство пользователей, но лишь немногие из них обладают достаточной квалификацией, чтобы разработать и прислать «заплатки» к программам для устранения определенных проблем с их применением. Опрос участников разработки ядра Linux показал, что большинство из них участвовало в создании системы в формах, не требующих специальных навыков программирования. Менее половины опрошенных было реально вовлечено в написание компьютерных программ. Только 30% из них прислало хотя бы по одной «заплатке» к программам в архив разработки и лишь 45% респондентов смогло сообщить о том, что ими написана хотя бы одна программная строка. Кроме того, только 45% опрошенных участвовало в проектах разработки отдельных подсистем, но 47% вообще не принимало участия в создании хотя бы одной подсистемы.

Анализ распределения исполнителей различных задач позволил выявить четыре категории участников разработки ядра Linux, образующих двухъярусную

структуру (табл. 9.2). Эта структура включает небольшой по численности «центр», в который входят лидер проекта и около сотни ответственных за отдельные подсистемы, и обширную «периферию», объединяющую собственно разработчиков и людей, присылающих сообщения об ошибках. В рамках двухъярусной структуры и протекает эволюционный процесс отработки операционной системы[34]. На «периферии» предлагаются и испытываются изменения к программам, из которых затем «центр» отбирает наиболее ценные для включения в систему. Если в традиционных моделях проведения НИОКР руководитель проекта сам принимает решения о том, что подлежит разработке и как внедрять полученные результаты, то в модели создания открытых систем эти решения разделены. На «периферии» сообщества разработчиков вырабатывают предложения по совершенствованию системы и приданию ей новых возможностей и характеристик, а в «центре» отбирают из поступивших предложений заслуживающие включения в официально рассылаемые версии системы, которые затем проходят многочисленные циклы доработок.

Таблица 9.2
Двухъярусная структура сообщества разработчиков ядра Linux, включающая четыре категории участников

Основные роли разработчиков Linux	Число участников данной категории	Суммарное число сообщений по e-mail от этой категории участников в архиве	Процент от общего числа сообщений в архиве
Лидер (руководитель) проекта		2840 (третье место по числу сообщений)	
Ответственные за подсистемы	1 121*	37 387 (включая сообщения лидера проекта)	**1,4**
Разработчики		20 563	
	2605**		18,8
			12,3
			10,3
Авторы сообщений об ошибках	1562"*	4216	**2,1**

* Установлено по списку имен, содержащихся в файле Maintainers, входящем в перечень файлов исходных кодов системы.

** Оценено по именам авторов сообщений, представленных в архиве e-mail и содержащих слово PATCH (заплата) в строке «Содержание сообщения».

*** Оценено по именам авторов сообщений, представленных в архиве e-mail и содержащих слово OOPS в строке «Содержание». OOPS — английское междометие, эквивалентное русскому восклицанию «Ой!» и выражающее удивление или испуг от ошибки. — *Примеч. пер.*

Источник информации: архив e-mail по ядру Linux за период с июня 1995 г. по август 2000 г.

«Центр». Основатель и бессменный лидер проекта создания операционной системы Linux Л. Торвалдс контролирует выпуск официальных версий системы. Именно ему принадлежит решающее слово относительно целесообразности включения того или иного дополнения или исправления в ядро Linux. Несмотря на мифы о его диктаторских замашках[35-37], Торвалдс, принимая ключевые решения, обычно консультируется с людьми, отвечающими за отдельные подсистемы, особенно в тех случаях, когда речь идет о проблемах, на решение которых тот или иной ответственный исполнитель потратил немало личного времени.

Согласно списку имен, содержащихся в файле Maintainers, за 132 подсистемы, составляющие ядро Linux, отвечает 121 человек, причем за некоторые подсистемы отвечают несколько человек. Ряд ответственных лиц закреплен за несколькими подсистемами одновременно. Анализ показывает, что закрепление ответственности за подсистемы происходит крайне неравномерно. Среди девяти модулей, входящих в версию Linux 2.2.14, за самый крупный из них — модуль драйверов устройств — отвечает 72% от общего числа ответственных исполнителей. Этот модуль включает 55% всех подсистем ядра Linux, на его долю приходится 58% объема ядра в мегабайтах и 63% программных строк. Следующий по размеру модуль вчетверо меньше модуля драйверов. По сравнению с программами, входящими в другие модули, драйверы устройств относительно просты в написании, но более сложны для отладки. Поэтому представляется оправданным, что большая часть усилий сообщества разработчиков ядра Linux сосредоточена именно на создании интерфейсов для тех периферийных устройств, которые каждый из разработчиков хотел бы подсоединить к своему компьютеру, особенно если учесть многообразие производителей и марок этих устройств, имеющихся на рынке и доступных для пользователей компьютеров.

«Периферия». Чтобы определить состав команды разработчиков, был просмотрен архив e-mail-системы за период с 1995 по 2000 г. и отобраны все авторы, приславшие хотя бы по одному сообщению, где в строке «Содержание» присутствовало слово PATCH. Предполагалось, что эти сообщения относятся к предлагаемым «заплатам» к программам, устраняющим выявленные ошибки или добавляющим новые свойства. В результате было выявлено 2605 человек, которых с полным основанием можно включить в состав команды разработчиков системы, существовавшей в 1995—2000 гг. Другую часть команды составляют люди, сообщающие о выявленных ими ошибках и недостатках программ. Они были определены путем отбора авторов сообщений, где в строке «Содержание» присутствовало слово OOPS. Задачи таких участников сводятся к выявлению ошибок, их описанию и подаче предложений по их устранению[38]. С 1995 по 2000 г. число авторов сообщений об ошибках составило 1562 человека, причем некоторые члены сообщества разработчиков входят в обе указанные группы одновременно. 45% людей, сообщавших об ошибках программ, также прислали по одному

сообщению с пометкой PATCH, а 29% авторов «заплат» — не менее одного сообщения с пометкой OOPS в строке «Содержание».

Существует распространенное мнение о том, что в процессе разработки OSS отсутствуют всякий контроль и ответственность[39-41]. При этом считается, что проблемы координации усилий разработчиков порождают задержки и хаос в процессе создания систем. Проведенный анализ тем не менее убеждает в наличии в проекте Linux как официальных, так и неформальных механизмов координации работ. Хотя крупномасштабные разработки обычно принято ассоциировать с сильной централизацией процесса принятия решений[42], в данном случае механизмы координации деятельности сообщества разработчиков ядра Linux устанавливают правила, согласно которым только принципиальные решения, такие как официальная рассылка версий системы, принимаются централизованно, а конкретные решения, касающиеся конструкции программ, принимаются на уровне отдельных модулей, но при полном контроле со стороны ответственных исполнителей за каждый модуль. Кроме того, процесс эволюционной отработки системы происходит в рамках двухъярусной структуры, «центр» которой отбирает доработки, создаваемые на «периферии», для включения в очередную официальную версию системы, а параллельно существует опытная версия, в рамках которой проверяются решения, имеющие новаторский характер.

Описанная структура доказывает, что сообщество разработчиков ядра Linux представляет собой систему, состоящую из многочисленных мелких модулей, причем каждый модуль функционирует в рамках естественной двухъярусной структуры. Такая система отличается высокой гибкостью. Каждый ее модуль представляет собой органичную подсистему, в рамках которой участник решает свои задачи «в пределах собственного понимания целей сообщества разработчиков системы в целом»[43].

Фред Брукс (Fred Brooks) выявил наличие сильной положительной корреляции между размерами организации и продолжительностью ее запаздывания с выходом на рынок с новыми предложениями. Эта зависимость получила название закона Брукса[44]. Анализ показывает, что эта корреляция может быть ослаблена, если поручать решение отдельных задач дополнительным исполнителям. Брукс считал, что сложность и стоимость коммуникаций внутри проекта растут пропорционально квадрату числа исполнителей, в то время как объемы выполненной работы увеличиваются с ростом числа исполнителей линейно. Однако в случае разработки ядра Linux появление в любом модуле большего числа людей, сообщающих об ошибках, никак не влияет на работу над другими модулями и не ведет к усложнению как самого модуля, так и системы в целом. Более того, применение Интернета и веб-приложений, благодаря которым обеспечивается многостороннее общение разработчиков, позволяет значительно сократить коммуникационные затраты. Понятно, что Брукс не мог прогнозировать появление подобных средств связи между участниками проекта. Таким образом, структурные особенности построения сообщества разработчиков ядра Linux, о которых шла речь выше, помогают ему успешно справляться с растущими масштабами и разнообразием решаемых задач.

Открытый эволюционный процесс отработки ядра Linux

Как уже отмечалось, двухъярусная структура сообщества разработчиков ядра Linux способствует реализации эволюционного процесса отработки системы, при котором «центр» сообщества отбирает или отклоняет многочисленные предложения по доработкам, поступающим с «периферии», включая наиболее полезные дополнения в очередную официальную версию системы.

Ключевым звеном, связывающим процессы выработки и отбора дополнений и изменений системы и придающим процессу ее отработки непрерывность и постепенность, служит рассмотрение и анализ предложений со стороны коллег. Как указано в файле Maintainers, «всякое, даже самое незначительное предложение должно быть обсуждено по крайней мере с четырьмя-пятью, а желательно и с большим числом специалистов». Рассмотрение предложений коллегами позволяет заменить контроль качества, когда ошибки выявляют в готовом продукте, их предупреждением на возможно более ранних стадиях. Такой подход хорошо согласуется с опытом организаций, добившихся высочайшего уровня качества своей продукции[45,46]. В процессе взаимодействия с коллегами происходит также коллективное обучение разработчиков. Таким образом, процесс отработки ядра Linux можно с должными основаниями считать процессом постепенного и непрерывного совершенствования продукта, который должен быть непременной особенностью любой самообучающейся организации, стремящейся к высокому качеству своей продукции и услуг.

Торвалдс считает, что процедура анализа любых предложений коллегами по сообществу разработчиков ядра Linux придает всему процессу отработки системы не только эволюционный, но также открытый характер[47]. Он указывает, что «распространенная ошибка заключена в участии в проверках программы ее составителей. В таких условиях анализ коллегами по сообществу лишен всякого смысла. Задача состоит в том, чтобы подобрать для проверки программ других людей, обладающих собственным опытом, отличающимся от того, каким обладают авторы программы».

Рассмотрение любых предложений коллегами представляет собой открытый процесс, в котором все предлагаемые изменения программ и их критика доступны для обсуждения и ссылок всем пользователям Интернета. Открытость процесса разработки системы позволяет членам сообщества, обладающим разным опытом и знаниями, анализировать любые предложения по изменению ее исходных кодов. Благодаря тому что любое предложение рассматривается многими разработчиками, его автор, который мог что-то упустить или не обладает

достаточным опытом для решения возникающих проблем, получает необходимую помощь в выявлении ошибок или других проблем, а также в повышении качества предлагаемых изменений к программам. Разнообразие уровня подготовки и опыта многочисленных разработчиков — это мощный инструмент для поиска, анализа и устранения ошибок в программах.

Одним из способов активизации анализа программ является увеличение частоты рассылки новых версий и сокращение продолжительности цикла разработки каждой из них. Чем скорее требуется получить заключения на новую версию системы, тем больше стимулов для разработчиков внести свой вклад в ее отработку. Таким образом, необходимость быстро отреагировать на новую версию системы поддерживает сообщество разработчиков в должном тонусе. Тем самым находит свое воплощение базовый принцип теории развития систем[48]. По сравнению с коммерческими ПП Linux представляет собой постоянно развивающуюся систему с очень частыми обновлениями ее версий (табл. 9.3). Большинство фирм — разработчиков коммерческих ПП объявляют о создании новых программ или доработанных версий существующих продуктов один раз в несколько лет, причем начало их массовой реализации зачастую задерживается, хотя в этих фирмах существуют ежедневно обновляемые отработочные версии каждого продукта, которые имеют хождение только внутри организации. (В отработочных версиях продукта ежедневно все файлы компилируются заново с учетом принятых изменений и объединяются в рабочую программу, которая затем проходит опытный прогон — относительно простую проверку, чтобы выявить отсутствие сбоев во время ее исполнения, — так называемый «smoke test».)

Процесс отработки системы Linux представляет пример непрерывного и постоянного совершенствования, причем ни одна версия системы никогда не становится окончательной. Во время одной из он-лайновых дискуссий вокруг ядра Linux в августе 1999 г. Л. Торваддс заявил, что считает благотворным подход, при котором «люди имеют возможность видеть все, что происходит», и настойчиво доказывал важность частой замены мелких «заплат» к программам новыми

Таблица 9.3 Хронология выхода основных официальных версий Linux

Версия	Первый и последний варианты версии	Даты выпуска первого и последнего варианта версии	Частота обновлений
1.0 1.2 2.0 2.2	Linux-1.0 Linux-1.2.0: Linux-1.2.13 Linux-2.0.0: Linux-2.0.38 Linux-2.2.0: Linux-2.2.16	12.03.94 06.03.95; 01.08.95 08.06.96; 25.08.99 25.01.99; 07.06.00	 14 раз за 5 месяцев 39 раз за 40 месяцев 17 раз за 18 месяцев
2.4	Linux-2.4.0-test1: Linux-2.4.0-test17	25.05.00; 23.08.00	7 раз за 3 месяца

версиями системы. Он, в частности, писал: *«Существо открытого процесса разработки заключено в том, что люди видят все, что происходит* (курсив автора). Не следует допускать, чтобы им был виден только конечный результат, скажем, через год. *Необходимо стремиться к тому, чтобы случайный человек имел возможность видеть любые, самые мелкие доработки системы — тогда, возможно, он сумеет избавить разработчиков от глупых ошибок*[49]. Ныне же, имея дело с громадными «заплатами» к программам, их пользователи ощущают беспомощность. Если же публиковать мелкие доработки по мере их поступления и иметь при этом веб-сайт, на котором их можно комментировать, люди получают возможность быть в курсе проводимых изменений системы и указывать мне и остальным разработчикам на те глупости, которые мы совершаем»[50].

В этом и заключена сущность модели непрерывного совершенствования, о которой столько говорят.

Анализ показывает, что руководитель проекта и ответственные за отдельные модули системы очень консервативно подходят к отбору предлагаемых исправлений и «заплат» к программам, включаемым в официальные версии системы. Только 23% опрошенных членов сообщества разработчиков ядра Linux подтвердили, что, по крайней мере, одно их предложение было реализовано. С одной стороны, такой подход обеспечивает сохранение устойчивости системы, но, с другой стороны, его эффективность может показаться достаточно низкой. Ведь 77% предложений оказывается пустой тратой времени и сил. Однако учитывая открытость процесса обсуждения всех предложений, те из них, которые были отклонены, создают питательную среду для обучения как самих авторов предложений, так и других членов сообщества. На этих примерах они получают возможность изучить те критерии, которым должны отвечать предложения, чтобы быть принятыми для дальнейшего совершенствования системы. Такая практика отбора формирует публичный форум, на котором . члены сообщества, анализируя замечания коллег, учатся тому, как следует совершенствовать собственную работу.

С момента выхода в свет первой версии Linux, как показывают подсчеты, в среднем каждую неделю появлялось по одной новой версии этой операционной системы. В табл. 9.3 и 9.4 приведена хронология появления официальных основных и экспериментальных версий ядра Linux, причем экспериментальные версии, выходящие параллельно с основными, обновляются чаще. Только в 1996 г. появилось 80 экспериментальных версий и примерно 30 основных версий системы.

Почти непрерывный поток версий Linux резко контрастирует со строго контролируемым и относительно редким выходом в свет новых версий коммерческих ПП. Это различие находит отражение в продолжающейся дискуссии вокруг латания брешей в системах защиты. Основные дебаты ведутся вокруг того, как часто следует доводить до широкой публики информацию об ошибках в ПП

Таблица 9.4 Хронология выхода экспериментальных версий Linux

Версия	Первый и последний варианты версии	Даты выпуска первого и последнего варианта версии	Частота обновлений
1.1 1.3 2.1 2.3 Pre-2.0 Pre-2.2	Linux-1.1.13: Linux-1.1.95 Linux-1.3.0: Linux-1.3.100 Linux-2.1.0: Linux-2.1.132. Linux-2.3.0: Linux-2.3.51 Linux-pre2.0.1: Linux-pre2.0.14 Linux-2.2.0-pre1: Linux-2.2.0-pre9	22.05.94; 01.03.95 11.06.95; 09.05.96 30.09.96; 22.12.98 11.05.99; 10.03.00 11.05.96; 05.06.96 28.12.98; 20.01.99	83 раза за 11 месяцев 101 раз за 11 месяцев 133 раза за 27 месяцев 52 раза за 10 месяцев 14 раз за 1 месяц 9 раз за 1 месяц
Pre-2.4	Linux-2.3.99-pre1: Linux-2.3.99-pre9	14.03.00; 23.05.00	9 раз за 2 месяца

Примечание. Обозначение версий системы и их вариантов производится с использованием иерархической системы нумерации, в которой первая цифра обозначает главную версию, а вторая — соответствующее ей дерево версий и вариантов. Основным версиям соответствуют четные вторые цифры в обозначении (например, 2.0; 2.2; 2.4), а экспериментальным — нечетные (например, 2.1; 2.3).

и заниматься их устранением. Поставщики коммерческих ПП утверждают, что они должны располагать достаточным временем для решения обнаруженных проблем, перед тем как делать их достоянием общественности, чтобы не информировать хакеров об уязвимости ПП. Однако специалисты по компьютерной безопасности считают целесообразным доводить подобную информацию до пользователей как можно раньше, чтобы те оказывали давление на поставщиков, заставляя их быстрее латать возникшие бреши в защите и устранять другие ошибки[51]. Применительно к Linux данная проблема стоит не столь остро, чем для коммерческих ПП. Безусловно, Linux тоже испытывает определенные сложности, обусловленные уязвимостью системы по отношению к действиям хакеров, но, по словам одного из пользователей, «сообщество разработчиков ядра Linux намного быстрее реагирует на проявления проблем безопасности, нежели поставщики коммерческих операционных систем. Выявление и латание брешей в защите Linux происходит публично и ведется намного быстрее»[52]. MITRE* идет в своих оценках еще дальше, утверждая в своем отчете, что способность быстро латать бреши, обеспечивая защиту против новых угроз кибернетического терроризма, «является отличительной чертой, присущей разработке ПП с открытыми исходными кодами, которая недоступна системам с закрытыми исходными кодами»[53]. Продукты с открытыми кодами обладают заметными преимуществами в смысле обеспечения их защиты, поскольку их исходные коды всегда доступны для тщательной проверки и анализа. Исходные коды коммерческих ПП хуже поддаются модификации[54].

Интернет-агентство полного цикла. — *Примеч. ред.*

Заключение

В настоящей главе приведены результаты анализа процесса разработки ядра операционной системы Linux, позволяющие выявить основные условия, обеспечивающие устойчивое поддержание ее высокого качества. Помимо общей схемы, присущей процессам создания OSS, существует ряд других факторов, позволяющих успешно разрабатывать ПП высочайшего качества с превосходными характеристиками. Существо процесса разработки OSS заключается в мобилизации и координации усилий многочисленных разработчиков, обладающих самыми разнообразными талантами, в рамках формализованных и неформальных организационных структур. Помимо этих структур, модульный принцип построения системы, использование двух параллельно существующих кодовых деревьев и двухъярусная структура сообщества разработчиков обеспечивают постепенное и непрерывное повышение надежности продукта и дополнение его новыми прогрессивными свойствами в ходе открытого эволюционного процесса рассмотрения и анализа вносимых предложений коллегами по сообществу разработчиков.

Ссылки и примечания

[1] Ядро операционной системы планирует решение различных задач на компьютере, включая исполнение пользовательских приложений, например, подключение к веб-браузеру, использо вание тестового редактора или СУБД, распределяет системные ресурсы компьютера между исполняемыми программами. Применительно к пользовательским приложениям ядро испол няет роль управляющей программы, обеспечивающей управление и планирование, межпро цессные переключения, управление устройствами ввода-вывода и оперативной памятью. При управлении аппаратными средствами ядро преобразует запросы оперативной системы в про граммы управления соответствующими устройствами с использованием их драйверов.

[2] The Internet Operating System Counter, http://www.leb.net/hzo/ioscount/index.html.

[3] Ricadela A. The Challenge That Is Linux // InformationWeek.Com. 6 May, 2002.

[4] Use of Open Source Software in the U.S. Department of Defense. Washington, D.C.: MITRE Corp. 2002.

[5] Hayes M. Quality First // InformationWeek.Com. 20 May, 2002.

[6] Ricadela A. Challenge That Is Linux 3.

[7] Ricadela A. Open for Business // InformationWeek.Com. 6 May, 2002.

[8] McDougall P. Businesses Turn to Open Source Systems as Vendors Add Offerings // InformationWeek.Com. 11 February, 2002.

[9] Garvin D. Managing Quality. New York: The Free Press, 1988.

[10] We are indebted to Terry Bollinger of MITRE Corp. for stimulating the following analysis.

[11] Эти данные приводит Грегори Х. Ватсон, консультант по качеству, бывший президент ASQ.

[12] See Juran J. M. and Godfrey A. B. Juran's Quality Handbook, 5th ed. New York: McGraw-Hill, 1999.

[13] Behlendorf B. Open Source As a Business Strategy in Open Sources: Voices from the Open Source Revolution, eds. DiBona, Ockman, and Stone. Sebastopol, CA: O'Reilly & Associates, 1999.

[14] Moon J. and Sproull L. Essence of Distributed Work: The Case of the Linux Kernel in Distributed Work, eds. P. Hinds and S, Kiesler. Cambridge, MA: MIT Press, in press.

[15] Markus M. et al. What Makes a Virtual Organization Work? // Sloan Management Review (2000).

[16] Zbaracki M. The Rhetoric and Reality of Total Quality Management. Ph.D. diss., Stanford University, 1994.

[17] Raymond E. S. The Cathedral and the Bazaar // First Monday: Peer-Reviewed Journal on the Internet (1998).

[18] Raymond E. S. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. Sebastopol, CA: O'Reilly & Associates, 1999.

[19] Hamilton D. Finally Microsoft Finishes Windows 2000 // Wall Street Journal, 16 December 1999, B12.

[20] See note 4.

[21] Исходные коды Linux открыты для свободного доступа и бесплатного скачивания с общедос тупного веб-сайта «Общественный архив Linux» по адресу http://www.kernel.org/pub/

[22] Torvalds L. The Linux Edge. In: Open Sources: Voices from the Open Source Revlution, eds. DiBona, Ockman, and Stone. Sebastopol, CA: O'Reilly & Associates, 1999.

[23] Только 28 ответственных за подсистемы перечислены в файле Credits.

[24] www.uwsg.indiana.edu/hypennail/ linux/kemel/

[25] Например, к августу 2000 г. было зарегистрировано 199 374 сообщений.

[26] www.linux.org/info/index.html.

[27] Нами было выявлено 14 535 человек, участвовавших в обсуждении и разработке ядра Linux, направивших за период с 1995 по 2000 г. хотя бы по одному e-mail-сообщению, представлен ному в списке рассылок. Поскольку число зарегистрированных пользователей Linux составля ет около 90 тыс., то, следовательно, по нашим оценкам 16% пользователей одновременно являются и разработчиками этой системы.

[28] See note 15.

[29] Raymond E. S. Homesteading the Noosphere // First Monday: Peer-Reviewed Journal on the Internet (1998).

[30] Lerner J. and Tirole J. The Simple Economics of Open Source // NBER (working paper, 2000).

[31] Raymond E. S. Homesteading. Section 6 on the hacker milieu as gift culture.

[32] See Veblen T. The Instinct of Workmanship and the State of Industrial Arts. New York: Viking Press, 1914.

[33] Cole R. E. Learning from Learning Theory: Implications for Quality Improvement of Turnover, Use of Contingent Workers, and Job Rotation Policies // Quality Management Journal 1. October, 1993.

[34] Lee G. and Cole R. E. The Linux Kernel Development: Cultural and Evolutionary Processes of Knowledge Creation (working paper, University of California-Berkeley, 2002).

[35] O'Reilly T. Ten Myths about Open Source Software. www.opensource.oreilly.com/news/myths_l 199.html (1999).

[36] Drakos N. Debunking Open Source Myths: Development and Support www.gartnerweb.com/public/ static/hotc/hc00088469.htmi.

[37] Drakos N. and Driver M. Debunking Open Source Myths: Origins and Players www.activestate.com / data/Gartner%20Group (2000).

[38] Между OOPS и ошибками ПП существует некоторое достаточно тонкое различие. Под ошиб кой понимают случай, когда что-то в системе, прежде всего в ядре, не работает так, как это следует при использовании определенного драйвера или реализации некоторого алгоритма ядра. Когда ядро операционной системы выявляет подобную неисправность, оно выдает сооб щение OOPS. Таким образом, OOPS представляет частный случай ошибки, поскольку ошибка может присутствовать и быть обнаруженной пользователем, но операционная система не вы дает сообщение о ней.

[39] See note 35.

[40] See note 36.

[41] See note 37.

[42] See Hage J. An Axiomatic Theory of Organizations // Administrative Science Quarterly 10 (1965).

[43] Bums T. and Stalker G. The Management of Innovation. London: Tavistock, 1961.

[44] Brooks F. The Mythical Man-Month: Essays on Software Engineering. Reading, MA: Addison-Wesley, 1995.

[45] Cole R. E. Managing Quality Fads. Oxford: Oxford University Press, 1999.

[46] Mizuno S. Company-Wide Quality Control. Tokyo: Asian Productivity Organization, 1992.

[47] Code Freeze; ISDN Perennial Lateness // Kernel Traffic, Aug. 3, 1999-Aug. 10, 1999 (44 posts): Re: no driver change for 2.4? http://kt.linuxcare.com / kernel-traffic / kt 19990819_31.epl#9

[48] Quoted from an e-mail sent by Linus Torvalds to the Linux kernel mailing list in August 1999.

[49] Steers R. and Porter L. Motivation and Work Behavior. New York: McGraw-Hill, 1991.

[50] LINUXCARE. Code Freeze: ISDN Perennial Lateness. Kernel Traffic (3 August 1999).

[51] Rosencrance L. Bug-Reporting Standard Proposal Pulled from IETF. Computerworld (21 March 2002).

[52] Weiss T. Users: Linux Security Flaws Won't Mar Linux. Computerworld (18 March 2002).

[53] MITRE, Use of Open Source Software, 53.

[54] Ibid.