Лицензионно-правовые особенности СПО

Юридические и правовые аспекты программного обеспечения

Понятно, что для организации "торговли воздухом" в виде передачи пользователю некоторых ограниченных прав на использование программного продукта за вполне конкретные деньги необходимо иметь право собственности на программный продукт. В англосаксонском праве, в Америке и большей части Европы этому соответствует понятие лицензии (англ. *license* --- разрешать, разрешение). Заметим, что до вступления в силу нового Гражданского Кодекса 2008 года в законодательстве РФ понятие лицензии отсутствовало, вместо этого использовалось понятие договора между разработчиком и потребителем.

Суть лицензии на творческий продукт состоит в следующем: при распространении этого продукта (объекта "интеллектуальной собственности") к нему прилагается то или иное предписание, в соответствии с условиями которого этот продукт распространяется и используется. Валидность этого предписания, его законодательная значимость, гарантирует соблюдение предписываемых правил.

"Несвободные" лицензии обыкновенно ограничивают возможности пользователя, свободные же --- дают ему те или иные "свободы". Типичная свободная лицензия включает в себя следующие свободы:

- Использование программы: пользователь может запускать программу и использовать ее результаты для любых своих целей. Заметим, что в текущем российском законодательстве данная свобода предоставляется пользователю автоматически: по факту получению программного продукта вас не могут заставить "что-либо не делать" (например, не использовать по субботам или воскресеньям).
- Изучение и модификация программного продукта. Когда Столман говорит "свобода", это значит, что ограничений в этом направлении быть не может. Данная свобода предполагает получение исходных текстов программы.
- Распространение программного продукта. Автор программного продукта не должен ограничивать как бесплатное, так и коммерческое его распространение.
- Распространение модифицированных версий. Эта свобода делает возможным организацию бизнес-модели на свободном ПО. Дело в том, что довольно часто можно встретить лицензии типа "я гениален, а вы не очень", вставляющие палки в колеса любому основанному на модификации программного продукта делу, например запрещающих продажу модифицированных версий. Наличие же данной свободы дает возможность зарабатывать деньги на внесении в продукт дополнительной, необходимой заказчику функциональности.

В свободных лицензиях группы GPL (General Public License) существует также дополнительное требование, не входящее в классическое определение свободного программного обеспечения (*Free and Open Source Software*). Это требование вызвано тем, что и разработчиками, и пользователям продукта вряд ли будет приятна ситуация, когда на базе свободного продукта можно будет сделать закрытый продукт. В истории программного обеспечения встречались ситуации типа "был академический свободный программный продукт, пришла компания, заплатила деньги за доработку или выкупила права у университета, и затем закрыла продукт". Разработчики часто чувствовали себя обманутыми, поскольку нанесен удар по академической среде, а свобода пользователей и разработчиков в итоге уменьшена. Для устранения этого противоречия и недопущения уменьшения свободы

пользователей Столман придумал специальное условие --- copyleft. Оно заключается в следующем: при модификации и распространении копий лицензия на модифицированный продукт должна быть "не хуже", чем исходная: она должна гарантировать все те же четыре свободы и сохранять условие copyleft. На практике это означает, прежде всего, свободный доступ к исходным текстам модифицированной программы. Отметим, что названные четыре свободы вместе с требованием copyleft и составляют лицензию GPLv2 (GNU General Public License version 2). Кроме "копилефт"-лицензий, существует множество свободных лицензий, не содержащих требования распространения модифицированых версий с исходными текстами, к ним относят лицензии BSD (старая и новая), MIT, Apache и многие другие.

Сделаем три замечания:

- Даже использование интерфейса GPL-библиотеки заставляет лицензировать программный продукт под не менее свободной лицензией. Данное требование для случая компоновки явно исключено, к примеру, в лицензии LGPL (GNU Lesser General Public License). Таким образом, LGPL-библиотеки защищены от создания своих закрытых модификаций, но могут использоваться программным обеспечением под иными лицензиями, в том числе и закрытыми.
- Существуют два несколько различающихся различных понятия: свободное ПО (*free software*) и ПО с открытым исходным кодом (*open-source software*). Определение ПО с открытым кодом по Реймонду (Eric Steven Raymond, ESR) состоит из десяти пунктов. Реймонд основал достаточно авторитетную организацию Open Source Initiative (OSI), которая определяет, является ли лицензия свободной. Это позволяет официально считать несвободными лицензии, предоставляющие доступ к исходным текстам, но, например, запрещающие продажу программного обеспечения.В рамках европейского законодательства понятия *free software* и *open-source software* можно считать эти понятия эквивалентными. Заметим, что в английском языке слово *open* более "сильное", чем *free* (которое часто означает "бесплатный", а не "свободный"), в русском же языке все наоборот: "свободный" и "открытый". В составленном для государственного проекта глоссарии оговаривается, что предпочтительный термин --- свободное ПО. Отметим также, что свободное программное обеспечение согласно определению не обязано быть бесплатным.
- Есть мнение, что в российском законодательстве нет возможности предписывать пользователю линию поведения. Соглашение может касаться лишь передачи прав собственности. С этой точки зрения, положение GPL о выпуске модифицированных продуктов под аналогичной лицензией выглядит незаконным (что не отменяет правомочности остальных ее положений), поскольку лицензия делает пользователя полноправным владельцем копии программного продукта по факту приобретения либо скачивания.

Наконец, несколько замечаний о практике применения в Российской Федерации свободного программного обеспечения с точки правоохранительных органов. Хотя в текущем законодательстве не требуется наличия лицензии в бумажном виде, это весьма желательно при проверке "лицензионности программного обеспечения". Поэтому дистрибутивы ПСПО содержат напечатанное "уведомление о правах" --- специальный документ, в котором перечислены предоставляемые пользователю дистрибутива права собственности. Кроме того, проверяющие органы могут потребовать "дистрибутив с серийным номером" и машину, "внутри" которой этот номер "установлен". В случае использования свободного ПО прямой возможности этого сделать нет по очевидным причинам. Тем не менее, даже в случае таких некомпетентных требований есть два способа подтвердить законность использования свободного программного обеспечения:

- предъявить упомянутое выше "уведомление о правах";
- предъявить купон технической поддержки с индивидуальным номером,

зарегистрированным на специальном сайте.

Такие документы поставляются, к примеру, с коробочными версиями ПСПО "Линукс Мастер" и могут в описанной ситуации помочь.

Сообщество свободного программного обеспечения

Структура сообщества вокруг свободного программного обеспечения --- очень важная тема, ведь именно рассматривая ее можно найти ответы на многие вопросы, касающиеся открытого процесса разработки. Итак, открытый способ разработки --- это то, чем изначально отличаются открытые (или свободные) программные продукты от закрытых, называемых также правовладельческими, собственническими или "проприетарными" (от англ. *proprietary* --- собственник). Все используемые при разработке технологии напрямую зависят от выбора способа разработки.

Совместная разработка

Открытый способ разработки плюс "копилефт" как способ лицензирования --- это основание для бизнеса, основанного на открытом программном обеспечении. Всякий человек, который воспользовался нашей разработкой и привнёс туда какие-то свои инструменты и который хочет достичь чего-то на рынке, обязан организовать этот бизнес также на основе свободного программного обеспечения, поскольку изначально разработка ведется под "копилефт"-лицензией. Таким образом, лицензирование методом copyleft --- это гарантия, что любые успешные модификации нашего программного обеспечения будут опубликованы, и ими сможет воспользоваться каждый, а развитие программного продукта не прекратится, даже если человек модифицировал что-то исключительно ради своего заработка.

Введение такой "заразной" свободной составляющей в лицензию приводит к тому, что, любой распространяющий свободный продукт производитель должен будет обеспечить свободный доступ к исходным текстам, и, стало быть, все сделанные им улучшения окажутся в свободном доступе. Поэтому любой автор программного продукта может смело опубликовать его под свободной лицензией с copyleft, зная, что, какие бы улучшения не были в него привнесены другими людьми, он может включить их обратно в исходный код. В принципе, это является одной из форм совместной разработки.

Более того, если какая-то компания вносит в продукт улучшения из коммерческих соображений, то для нее часто проще делать это сразу открыто, используя доступ к первоначальному исходному коду, потому что согласно лицензии ей все равно необходимо будет опубликовать исходный код, если она намеревается распространять этот продукт. Такая ситуация имеет место не со всеми открытыми программными продуктами, в ряде случаев измененная версия разрабатывается и ее исходный код распространяется отдельной фирмой. Тем не менее, подход, основанный на непосредственной совместной разработке, делает распространение и улучшение программ более быстрым и простым. Сейчас так происходит далеко не везде, но, например, в случае с IBM и Samba, или с ядром Linux ситуация именно такова: разработчики из различных и часто конкурирующих коммерческих фирм работают с одной и той же системой контроля исходного кода свободной программы.

Программы GNU/Linux --- пример совместной разработки

Мы подобрались к вопросу о том, что такое операционная система GNU/Linux, как она разрабатывается и кто и как ее использует. Все элементы этой огромной системы, будь то прикладное приложение, ядро операционной системы, или документация, разрабатываются отдельным человеком или отдельной командой разработчиков, и это фактически единственная возможность для подобных разработок, общая трудоемкость которых исчисляется десятками тысяч человеко-лет. Таким образом, у каждого элемента системы есть некий основной автор, но этим круг людей, причастных к разработке этого элемента, не ограничивается.

Итак, существуют несколько человек (иногда даже только один), образующие основную группу разработчиков (англ. *core team*). Это, как правило, те люди, которые большую часть своего времени тратят на разработку этого продукта. Они могут быть сотрудниками одной фирмы, а могут и жить в разных странах. Иногда они зарабатывают этими разработками себе на хлеб, а иногда это дело жизни. Если бы этим всё и ограничивалось, то ситуация бы ничем не отличалась от той, что имела место в связи с правовладельческими продуктами, потому что любой программный продукт разрабатывается некой командой. Правда, функция основной группы разработчиков шире, чем функция разработчиков правовладельческого продукта, поскольку обычно они же занимаются ещё и стратегическим планированием развития продукта.

В отличие от разработчиков правовладельческого продукта основная группа разработчиков рассчитывает на совместную разработку, на привлечение к сообществу любых людей, в том числе тех, которые не могут уделять разработке нашего продукта всё своё время. Тем не менее, в сообщество входят и профессиональные программисты, перед ними периодически возникает задача внести в наш продукт какие-то изменения, допустим, исправление ошибок или внедрение новых возможностей для своих целей или целей своего работодателя.Таким образом, подключаются добровольцы, которых может быть достаточно много (это зависит от популярности проекта). Если доброволец исправил единственную ошибку, и основная группа разработчиков это приняла, то он уже является разработчиком. Эти люди заинтересованы в программном продукте (например, получают зарплату за его внедрение), имеют пользовательскую квалификацию и квалификацию разработчика, имеют временные ресурсы на участие в сообществе. Откуда у добровольца эти ресурсы, и что конкретно им двигало --- это важно для исследователя, но не регламентировано.

Если изменения от разработчиков-добровольцев попадают в общий исходный код (англ. *upstream*), то им полагается огромное спасибо от разработчиков и пользователей. Таких людей больше, чем основных разработчиков, но никаких дополнительных обязательств перед ними у них нет. С другой стороны, они должны быть достаточно мотивированны, и если кто-то однажды принял участие в модификации продукта, вполне вероятно, что он сделает это снова, и даже может быть принят в основную группу разработчиков.

Рассмотрим теперь роль пользователей в сообществе. Обратите внимание, что их должно быть ещё больше, чем разработчиков. Это необходимо по нескольким причинам. Так, любой разработчик имеет опыт и как пользователь, и как разработчик, и он знает, как работает та или иная часть программного продукта, даже если она никак не документирована. Однако при таком подходе продукт очень плохо функционирует с точки зрения конечного пользователя, не являющегося программистом. На пользователей никаких обязанностей не накладывается, единственное, что надо иметь в виду --- пользователь хорош тогда, когда он активен. Активность проявляется в том, что пользователи заявляют разработчику, что им неудобно пользоваться программным продуктом. В этом случае социальное чувство (или опасения лишиться дохода) заставит разработчика задуматься над тем, как сделать, чтобы менее квалифицированным пользователям было тоже удобно. Далее, часто документация бывает неполной, и пользователю нужен совет разработчика, что является причиной задуматься о полноте документации. Наконец, при нахождении любых ошибок пользователю хорошо было бы о них сообщать разработчикам, чтобы они могли внести исправления. Наконец, наиболее активные пользователи могут сами участвовать в создании документации или в ответах на вопросы менее квалифицированных пользователей. Таким образом, сообщество состоит из разработчиков программного продукта и его активных пользователей.

У описанного процесса свободной разработки есть обратная сторона. Разработчики должны что-то делать. То есть, если у вас есть идея, как модифицировать программный продукт, будьте любезны, модифицируйте. В случае свободного программного обеспечения все возможности в руках пользователей и разработчиков. Если активности участников процесса разработки не будет, то и развития не будет, причём это касается как разработчика-

программиста, так и пользователя. Типичный случай продукта, не соответствующего этой схеме, это продукт, у которого очень мало активных пользователей. То есть, есть основная группа разработчиков --- разработчики, которым за этот продукт заплатили, --- и есть, скажем, три пользователя. В такой ситуации развития не будет, потому что нет привлечения широких слоев пользователей к процессу, и не будет мотивации у разработчиков для улучшения программного продукта.

Разработка дистрибутива GNU/Linux

Из описанного процесса разработки отдельных элементов системы GNU/Linux может сложиться впечатление, что дистрибутив --- это набор разрозненных программных продуктов, непонятным образом собранных вместе. То есть, какие-то люди разрабатывают собственно GNU/Linux, и совсем другие люди занимаются разработкой программ. На самом деле, дистрибутив можно рассматривать как метапродукт, который подчиняется тем же законам сообщества, которым подчиняется отдельный продукт.

Дистрибутив --- это тоже программный продукт, только очень и очень большой. Он собирается из частей, которые находятся в интернете под свободными лицензиями. Специфика дистрибутива в том, что он многокомпонентен, причем компоненты его разного размера и сложности. Если мы пишем программу совместно, там может быть 30 мегабайт исходного кода, множество отдельных модулей, но даже если мы возьмём программу небольшого размера, то это всё равно некоторое единое целое, с единым представлением, как следует программировать, документировать и т.д. Дистрибутив GNU/Linux --- это набор нескольких тысяч программных продуктов, сделанных, как минимум, сотнями групп разработчиков, часто никак не связанных друг с другом. Поэтому функции разработчика дистрибутива можно сформулировать следующим образом: определение политики развития дистрибутива, а также целевая разработка программных продуктов, которые сообществу в целом не очень нужны. Кстати, здесь и один из ответов на вопрос, откуда берутся деньги при открытом способе производства: один из неплохих источников --- заказная доработка для нужд конкретного заказчика.

Рассмотрим сообщество разработчиков с позиции разработчика дистрибутива. В принципе, включаемые в дистрибутив программные продукты разрабатываются и без его участия, и он может просто зайти на сайт и их скачать. Проблема в том, что нежелательно скачивать бинарные файлы (это приходится делать в двух случаях: если вы совсем не разбираетесь в GNU/Linux, а вам необходимо запустить такую программу, или если этот очень нужный вам продукт распространяется вообще без исходного кода). Скачивать бинарные файлы не стоит по трём следующим причинам:

- Технологическая несовместимость. Эта программа может просто не заработать в этой системе (например, у нас может быть другая архитектура процессора --- x86-64, а на x86).
- Если в бинарном файле нашлась некоторая ошибка или уязвимость с точки зрения безопасности, с бинарным файлом мы сделать ничего не сможем.
- Программа может работать нормально, но нарушать какую-то дисциплину, установленную разработчиками дистрибутива. В случае ее включения в дистрибутив в неизменном виде всё остальное может начать работать некорректно.

Кто будет заниматься всеми этими проблемами, то есть отслеживанием новых версий, сборкой программных продуктов из исходных версий в соответствии с политикой дистрибутива, адаптацией (в случае, если есть какие-то ошибки), оперативным исправлением уязвимостей? Это ментейнер --- человек, который играет роль разработчика по отношению к дистрибутиву. Он берёт у разработчиков программу в исходных текстах, чтобы проверить на предмет отсутствия ошибок и на предмет соответствия политике дистрибутива, затем изготовляет из нее бинарные версии для всех поддерживаемых дистрибутивом аппаратных

архитектур, а затем и взаимодействует с пользователями дистрибутива. Он отличается от разработчика исходной версии тем, что он хорошо разбирается в данном программном продукте, который он собрал и адаптировал к использованию в дистрибутиве. Ряд изменений, сделанных ментейнером, может быть включен в дальнейшем в исходный код продукта (в "апстрим").

Таким образом, структура сообщества дистрибутива копирует структуру сообщества, нарастающего вокруг обычного программного продукта. Основная группа разработчиков принимает стратегические решения и определяет политику дистрибутива и требования к составляющем его программам. Разработчик в данном случае --- это тот, кто программу не пишет, а адаптирует её для работы в дистрибутиве, это могут быть как разработчики-добровольцы, так и члены основной группы разработчиков. Пользователи --- это в первую очередь пользователи дистрибутива как такового --- могут быть и системными администраторами, внедряющими и поддерживающими дистрибутив на предприятии, и обычными домашними пользователи.

И в случае дистрибутива никто от моральных обязательств не освобожден. Основная группа разработчиков дистрибутива принимает стратегически верные решения, дело отдельного ментейнера --- квалификация и слежение за продуктом. От пользователя, как и в предыдущем случае, требуется активность. Так, в случае прямого обращения к ментейнеру некоторые задачи решаются в течение часов даже в случае некоммерческих дистрибутивов.

Помимо упомянутых выше обязанностей членов сообщества существуют также другие условия, которые необходимо соблюдать для комфортного существования сообщества:

- Информационная связность. Иными словами, свободный обмен информацией через интернет. Не должно быть никаких препятствий на пути распространении информации, причём как самого программного продукта, так и информации о нём.
- Информационная структурированность. Речь о том, что информации должно быть не просто много, но она должна быть хорошо структурирована, чтобы человек мог подключиться к проекту с минимальными затратами. При этом необходимо понимать, что в понятие информационной структурированости включается как чисто техническая информация для разработчиков, так и пользовательская документация. С другой стороны, структурированность не является основной целью, и далеко не всегда её удаётся поддерживать на должном уровне. В эту же категорию входят интерактивные ресурсы --- обратная связь по ошибкам, списки рассылки, форумы и так далее. Их существование необходимо для того, чтобы пользователь, у которого есть вопрос, знал, кому и как его задать.
- Технологические преимущества. Смысл в том, чтобы человеку из команды был удобно не просто компилировать, а компилировать в соответствии с принятой дисциплиной. То есть, необходимо предоставить сборочные сервера, автоматические механизмы сборки и тестирования, ведение учёта изменений, поддержку версионности. Иными словами, предоставить то, что облегчит ему жизнь в сообществе.

Последний пункт можно пояснить на прмере. Предположим, мы скачали какую-то программу на языке Си, в ней около ста файлов. Она была разработана, скажем, под ОС Sun Solaris и архитектурой Sparc64, и там у автора всё замечательно работает. А мы хотим скомпилировать её под GNU/Linux и архитектуру x86, поскольку других подобных программ нет. Сначала мы определяем, какие библиотеки и каких версий требуются для ее компиляции и работы. В итоге ментейнер собирает ее, дает команду `make install`, но в Solaris'e программы находятся в каталоге `/opt/progname/`, и ментейнер вынужден всё удалить, и дальше уже вносить изменения, чтобы она помещала свои файлы в правильные каталоги в соответствии с политикой дистрибутива, и там их и искала. Затем оказывается, что есть какая-то ошибка, которая у автора не проявляется в силу иной архитектуры и операционной системы. После ее исправления, наконец, наступает момент, когда программа работает нормально, но через 2

недели выходит ее новая версия... Чтобы избежать ситуации, когда этот процесс придется повторять каждый раз с нуля при выходе новой версии, необходимо помнить следующее: любые изменения необходимо протоколировать , кроме того, нужно указывать, что нужно для сброчного окружения. И только после этого можете спокойно это собирать.

Для определения различий между исходной и измененной версиями этого существуют специальные программы diff и patch: первая показывает различия между старым и новым файлом (эти различия называются "патчем"), а вторая изменяет содержание старого файла на содержание нового. При выходе новой версии созданный ранее патч применяется к тексту программы. Если патч не накладывается, то, вероятно, автор либо исправил ошибку сам, либо внес крупные изменения в тот фрагмент кода, где была ошибка. Ментейнеру нужно выяснить, что же там изменилось, возможно, ошибка уже исправлена, или нужно создавать новый патч. На жаргоне весь этот процесс называется "накатить патчи". Осталось сказать, что делать это лучше на отдельном сборочном компьютере, чтобы избежать возможных нежелательных изменений библиотек на своем компьютере

Открытый и закрытый процессы разработки

Каковы же были причины раскола в среде разработчиков, разделившего все существующее в настоящий момент программное обеспечения на свободное и несвободное?

С одной стороны, еще до формирования понятия "свободное программное обеспечение" в академической среде возник весьма эффективный способ разработки, который приводил к созданию программ, удобных для пользователей-программистов: каждый мог участвовать в разработке и изменять разрабатываемую программу. Первоначально история UNIX-систем была основана на том, что люди писали и дописывали программное обеспечение для своих нужд. Смысл свободного программного обеспечения, по Столлману, как раз и состоит в том, что любой, кто хочет что-то сделать с программой --- может это сделать. Это приводило к развитию программного обеспечения в той форме, которая была удобна тем, кто с ним работает.

С другой стороны, при таком способе разработки исходный код программы не является скрываемым от посторонних глаз объектом собственности, и возможность его свободного копирования не позволяла зарабатывать столь же много денег на разработке программного обеспечения, как позволила бы закрытая модель. Совершенно очевидно, и это явственно наблюдалось в пресловутом открытом письме Гейтса, что если исходный текст программы не является секретным, а все копии программного продукта не признаются объектом собственности автора программы, то очень много денег на программном обеспечении заработать достаточно проблематично, в отличии от случая, когда программное обеспечение и все его копии объявляются объектом собственности автора --- в частности, их нельзя копировать или сдавать аренду без разрешения правообладателя.

Причиной существования двух точек зрения на программное обеспечения является то, что программа, по своей природе --- нематериальный объект, обладающий свойством безущербного копирования. Суть безущербного копирования состоит в следующем. Для создания материального объекта требуются значительные переменные издержки: сырье, трудовые и энергетические затраты, причем для производства нескольких предметов все эти затраты часто можно примерно считать равными затратам на производство одного, умноженным на количество предметов. Для нематериальных объектов, к которым относятся и программные продукты, при производстве нескольких копий продукта затраты примерно такие же, как и при производстве одной копии продукта. Такая же ситуация с большинством творческих произведений -- многие из них легко тиражируются, особенно в цифровой форме.

Обратите внимание на три следствия, вытекающих из нематериальной природы программного обеспечения.

- При производстве программных продуктов, в отличие от объектов материальной собственности, переменные, относительно числа копий, издержки --- малы. Акт копирования программного обеспечения практически ничего не стоит. Таким образом, при больших тиражах нематериального объекта сырьевыми и энергетическими затратами можно пренебречь. Более того, и энергозатраты, и затраты на расходные материалы можно свести практически к нулю, если продукт распространяется по сети общего пользования.
- В противовес этому, трудозатраты на создание таких нематериальных объектов могут быть очень велики, в особенности, если это творческий продукт --- в этом случае измерить их практически невозможно.
- На этой особенности можно построить очень выгодный бизнес. Нужно только скрыть от внимания потребителей возможность безущербного копирования, сказать, что это объект собственности и сказать: "мы правообладатели этого продукта и всех его копий, мы определяем политику его распространения". После чего при большом

объеме продаж мы можем получать прибыли, измеряющиеся в тысячах процентов, которые каким-то образом перераспределяются внутри компании-правообладателя. Мы имеем дело с принципиально новым видом собственности, который не измерим по правилам, которые применяются к материальным объектам.

Если исходить из радикальной позиции, то деньги в случае программного обеспечения и подобных объектов --- неприменимый способ измерения стоимости такого продукта. Возможно, что позиция, заключающаяся в том, что все программное обеспечение должно быть свободным, и что нельзя брать деньги за копирование, может быть даже экономически оправдана. Существование несвободного программного обеспечения не только позволяет создать бизнес с потенциально неограниченной нормой прибыли, но и все больше ставит под сомнение деньги как эквивалент товаров и услуг.

Учитывая возможности построения на производстве программного обеспечения бизнеса с потенциально огромной нормой прибыли, может показаться странным, что свободное программное обеспечение вообще существует. Кажется, что его разработкой должны заниматься фанаты, которые готовы потерять возможность получения сверхприбыли ради несомненно приятной, но абсолютно безденежной идеи совместной работы и открытого обмена идеями. Тем не менее примеры многих компаний, построивших свой бизнес на открытом программное обеспечении, вполне показательны. В чём же технологические потери при переходе на закрытый способ разработки?

- Открытый способ разработки придерживается следующего правила --- копирование не причиняет ущерба разработке. Из неограниченного рамками закона копирования производитель можно извлечь определенную выгоду. Выгода от неограниченного копирования в том, что чем больше людей будут знать о программе, тем больше людей будут иметь возможность ею воспользоваться, тем больше людей захочет её улучшить и тем больше вероятность того, что найдётся тот, кто действительно это сделает или предложит, как это сделать. Таким образом, свободное программное обеспечение позволяет привлечь более широкий круг пользователей, тестеров и даже эпизодических разработчиков. Поэтому круг разработчиков свободного программного обеспечения может быть очень широким.
- Разработчикам свободного программного обеспечения крайне выгодно распространение не только самих программных продуктов, но и полной информации о них. Вокруг открытых программных продуктов может быть создано открытое информационное пространство, поскольку никакая часть информации о программном продукте не является коммерческой тайной, включая его исходные тексты. Следует отметить, что организация информационного пространства это весьма сложный процесс: информации не только должно быть как можно больше, кроме того она должна быть как можно более разумной и структурированной, должна быть рассчитана в первую очередь на то, что ей воспользуются.

Итак, что же теряют разработчики теряем, переходя на закрытый способ разработки?

- Законодательно, для того, чтобы кто-то мог написать ограничивающее лицензионное соглашение о программном продукте, он должен оставаться хозяином всех копий программного продукта. Покупатель получает некоторые ограниченные права на использования программного продукта, но при этом не получает никаких прав собственности на него: основная фраза в несвободной лицензии на программное обеспечение --- "this software is licensed, not sold". Полным собственником всех копий программного продукта, организованных и разрабатываемых по закрытому способу, обязана оставаться некая единая организация. А уже это накладывает на весь процесс сильные технологические ограничения.
- Главное ограничение состоит в том, что, вообще говоря, мы не можем позволить утекать информации, особенно исходным текстам программного продукта.

> Информация становится главной коммерческой тайной и оберегаемой "интеллектуальной собственностью". Коммерческая тайна и "интеллектуальная собственность" становятся управляющим механизмом бизнес-процессов. Невозможно допустить ситуацию, когда наши конкуренты узнают о нашем программном продукте достаточно информации, чтобы воспроизвести его и начинать брать за копирование аналогичного программного решения на, допустим, 20% процентов меньшее денег.

Таким образом, большие деньги можно сделать только если всё, что может быть воспроизведено путём очень дешёвого копирования, держится в тайне, и защищается от копирования законодательством. В первую очередь это исходный текст программного продукта. В результате из-за ограничений в информационном плане обычно теряется качество программных продукта, поскольку открытое информационное пространство желательно для любого программного продукта. Любому продукту нужна реклама, нужно распространение информации о себе, нужна возможность изучения классифицированными пользователями для выявления ошибок и особенностей функционирования. Но в случае закрытой разработки информация, становящаяся открытой, жестко ограничивается. Не допускается разглашение того, что называется коммерческой тайной, либо предоставление доступа к этой информации возможно только на коммерческих условиях или условиях неразглашения. Например, преподаватели ВУЗов в последнее время теоретически могут получить ограниченный доступ к исходным текстом ядра ОС Windows --- на условиях довольно сложной и ограничивающей их лицензии. Неэффективными для создания открытого информационного пространства оказываются даже лицензии, подобные Microsoft Shared Source Reference License, запрещающие программирование с использованием взятой из исходных кодов информации для всех целевых платформ кроме некого множества.

В заключении приведем сравнительную таблицу, подытоживающую два подхода к разработке программного обеспечения.

Открытый способ разработки	**Закрытый способ разработки**
Копирование не наносит ущерба разработке и способтвует рекламе продукта	Копирование считается наносящим убытки, для распространения информации о продукте создаются демо-версии и негласно допусается некоторый объем пиратских копий
Совместная разработка: может присоединиться любой полезный человек	Корпоративная (закрытая) разработка, ограниченный круг участников
Открытое информационное пространство с точки зрения как объема, так и доступности информации	Ограниченное информационное пространство: либо по объему, либо есть социально-технологические ограничения для доступа
Нет "хозяина всех копий"	Существует единственный собственник всех копий

Таким образом, закрытый способ разработки обычно увеличивает издержки на тестирование, сохранение коммерческой тайны, расходы на предотвращение заимствования решений конкурентами и недопущение нелегального копирования программного продукта пользователями. С точки зрения большинства разработчиков программных продуктов, закрытый подход выглядит менее интересным способом существования. Раньше считалось, что при закрытом подходе жить хлебнее, но на сегодняшний день зарплаты разработчиков в компаниях, ведущие бизнес на открытом программном обеспечении, принципиально не

отличается от зарплат при закрытом подходе разработки, как показывают независимые исследования.