Почему у Свободного ПО плохое юзабилити и как его улучшить

Мэтью Пол Томас (Matthew Paul Thomas)
Перевод: Дмитрий Стасюк <wintermute@undenied.ru>

11 августа 2008 г.

Источник: http://mpt.net.nz/archive/2008/08/01/free-software-usability

Когда я писал первую версию этой статьи 6 лет назад, я назвал ее "Почему юзабилити Свободного ПО сдвигается в худшую сторону". Лучшие программы с открытыми исходными кодами и операционные системы сейчас более пригодны к использованию, чем были тогда. Но эти улучшения по большей части происходят из-за медленно нарастающих улучшений и незначительного соревнования между проектами и дистрибьюторами. Основные проблемы в самом дизайне остаются, по большей части, не исправленными.

Многие из этих проблем относятся к любительскому ПО вцелом, а не к Свободному ПО в частности. Написанные любителем собственнические программы зачастую трудно использовать по множеству тех же причин. Но наиболее легкий способ заставить добровольцев вносить свой вклад в программу - открыть исходные коды. И в то время как основная работа тысяч людей связана с разработкой Свободного ПО, большинство его разработчиков - добровольцы. Поэтому именно в Свободном ПО проблемы юзабилити любительских программ встречаются наиболее часто.

Это дает нам ключ к нашим первых двум проблемам.

1. Слабые мотивы для улучшения юзабилити. Производители собственнического ПО, обычно, зарабатывают деньги создавая программы, которые люди хотят использовать. Это сильный стимул сделать его более пригодным к использованию. (Это работает не всегда: например, программы Microsoft, Apple и Adobe иногда становятся хуже, но продолжают доминировать благодаря сетевому эффекту. Но большую часть времени это работает.)

С проектами добровольцев, тем не менее, любая мотивация намного слабее. Количество пользователей едва ли дает какое-либо финансовое изменение разработчикам, да и для свободно распространяемого ПО практически невозможно подсчитать число пользователей. Существуют другие мотивы - произвести впечатление на будущих работодателей, или добиться включения своей программы в дистрибутив популярной ОС - но они, скорее, косвенные.

Решения: Увеличить количество и силу мотивирующих факторов. Например, ежегодные награждения дизайна Свободных программ могли бы публиковать и награждать разработчиков за хороший дизайн. Распространители (дистрибьюторы) программ могли бы публиковать статистические данные о том какое количество их пользователей использует какие программы, и как это число изменяется во времени. Система поощрений могла бы позволить людям платить деньги по факту того как кто-либо внесет конкретные улучшения юзабилити. А распределенный контроль версий поощрял бы быстрое соревнование: дистрибьюторы могли бы выбрать не только какое приложение поставлять, но также и ветвь приложения с лучшим, по их критерию выбора, юзабилити.

2. Несколько хороших дизайнеров. Некоторые музыканты являются также отличными композиторами, но большинство не такие. Некоторые программисты также замечательные дизайнеры, но большинство - нет. Программирование и дизайн пользовательского интерфейса - разные знания, и люди, преуспевшие в обоих, встречаются редко. Так что это важно - иметь выделенных дизайнеров для программы, но немногие проекты Свободного ПО их имеют. Некоторые специалисты по юзабилити приняты на работу разработчиками Свободных программ такими как Mozilla, Sun, Red Hat и Canonical. Но таких немного, а опытных добровольных дизайнеров найти еще труднее.

   Решения: Предоставлять высокодоступные учебные материалы для программистов и добровольных дизайнеров для улучшения общего уровня знаний в области дизайна. Поощрять сообщества, которые дают возможность программистам сотрудничать со специалистами по юзабилити. А также подталкивать Свободные программные проекты к тому чтобы ведущий программист, ведущий дизайнер пользовательского интерфейса, редактор справки и инженер контроля качества были разными людьми.

Но почему нехватка добровольных дизайнеров стоит на первом месте? Это приводит нас к третьей проблеме.

3. Предложения дизайна часто не запрашивают или не приветствуют. Свободное ПО имеет долгую и здоровую традицию "покажи мне код". Но когда кто-то указывает на проблему юзабилити эта традиция превращается в "патчи приветствуются", которая бесполезна поскольку большинство дизайнеров не программисты. И не очевидно как еще специалисты по юзабилити могут выручить.

   Решение: Создать процедуру, по которой специалисты по юзабилити могут вносить свой вклад в проект. Например, ведущий дизайнер мог бы публиковать параметры дизайна на веб-сайте проекта и привлекать обратную связь в блоге, вики или списке рассылки. Дизайнер мог бы вежливо ответить на предложения дизайна (даже на неправильные). А координатор (мэйнтейнер) проекта мог бы установить редактируемую систему отслеживания проблем вместо системы отслеживания ошибок (bug tracker), что сделает более легким улучшение, прием и отклонение, приоритезацию реализации предложений дизайна тем же способом что и отчеты об ошибках.

Так почему же программисты по разному реагируют на предложения дизайна отдавая предпочтение техническим отчетам об ошибках?

4. Юзабилити трудно измерить. Некоторые показатели программ легко и точно измеряются: работает ли она вообще, как быстро она запускается, корректна ли она с технически.

   Но это лишь частичная замена более важных показателей, которые труднее измерить: полезна ли программа, насколько отзывчивой она ощущается, ведет ли она себя так как этого ожидают люди, какая часть людей успешно ее использует, как быстро они могут ее использовать, насколько они удовлетворены в конце.

   Эти, относящиеся к человеку, показатели зачастую могут быть измерены в пользовательских тестах. Но проведение таких тестов занимает часы или дни, которые добровольцы не хотят тратить. Пользовательские тесты обычно не очень точны, выхватывают лишь крупные проблемы, но оставляют дизайнеров без твердых оснований для убеждения программистов относительно небольших проблем. И даже если проблема определена, решение необходимо разработать, и оно может также потребовать тестирования.

   Без частого пользовательского тестирования проекты добровольцев полагаются на субъективную обратную связь от категории людей достаточно преданной чтобы подписаться на список рассылки проекта. Но то что говорят эти люди не может отражать даже их

собственное настоящее поведение, не говоря уже о поведении пользователей вообще.

Решения: Продвигать технологии мелкомасштабного пользовательского тестирования, осуществимые для добровольцев. Разработать и продвигать программы для захвата содержимого экрана, видеозаписи и других программ, которые облегчают проведение тестов. Стимулировать разработчиков доверять результатам пользовательского тестирования больше чем мнениям пользователей. И написать руководства по дизайну, дающие советы по решению общих мелких проблем, которые пользовательские тесты не могут выловить.

Нехватка выделенных дизайнеров, в свою очередь, способствует появлению трех культурных проблем в проектах Свободного ПО.

5. Кодирование до дизайна. Программа имеет тенденцию быть более удобной в использовании если, хотя бы в черновом варианте, был сделан дизайн до написания кода. Желаемый пользовательский интерфейс для программы или функции может затронуть модель данных, выбор алгоритмов, последовательность, в которой выполняются действия, необходимость использования нитей (threads), формат хранения данных на диске, и даже набор функций программы вцелом. Создание всех этих скелетов и прототипов кажется скучным, поэтому программист часто просто начинает писать код - он позаботится об интерфейсе позднее.

   Но чем больше написано кода, тем сложнее исправить проблемы дизайна. Так что программисты предпочитают не заморачиваться или убеждают себя в том что это, на самом деле, не проблема. И если они, наконец, исправят интерфейс после версии 1.0, уже существующие пользователи должны будут освоить его снова, что расстроит их и подтолкнет к рассмотрению конкурирующих программ.

   Решение: Объединить в пары дизайнеров и тех программистов, которые хотят разрабатывать новый проект или новую функцию. Ввести в Свободное ПО культуру: сначала дизайн, потом код.

6. Слишком много авторов. В отсутствие выделенных дизайнеров множество участников проекта пытаются внести свой вклад в дизайн пользовательского интерфейса независимо от того насколько они знакомы с предметом. Множество дизайнеров ведет к разобщенности, и в общей концепции и в деталях. Качество дизайна интерфейса обратно пропорционально числу дизайнеров.

Решение: У проектов могут быть ведущие дизайнеры пользовательского интерфейса, которые принимают предложения всех остальных и работают с программистами над принятием решения о реализуемости этих предложений. Также руководства и более детальные спецификации дизайна могли бы помочь избежать свойственных программистам слабостей.

7. Гонка за лидером. В отсутствие собственного четкого дизайна многие разработчики предполагают что какая-нибудь Microsoft или Apple создали хороший дизайн. Иногда так и есть, но иногда - нет. Имитируя их дизайнерские решения разработчики Свободного ПО повторяют их ошибки и убеждаются в том что у них никогда не будет лучшего дизайна, чем у собственнических альтернатив.

   Решение: Поощрять новаторский дизайн посредством наград и других форм прославления. Обновить руководства по дизайну, где это нужно, для отражения результатов успешных дизайнерских экспериментов.

Другие причины низкой юзабилити существуют независимо от существования выделенных дизайнеров. Эти проблемы труднее решить.

8. У кого что болит ... Добровольные разработчики работают над проектами и функциями, которые им интересны, что обычно подразумевает программы, которые они собираются использовать сами. Будучи разработчиками программ они также являются опытными пользователями. Так что эти программы, как они их себе видят в повседневном использовании, получаются слишком необычными и сложными. Функции, необходимые скорее новым или неподготовленным пользователям, такие как родительский контроль, помощник в настройке или возможность импортировать настройки из соперничающего ПО могут быть забыты или совсем не реализованы.

   Решения: Ввести культуру простоты восхваляя сдержанный дизайн и высмеивая дизайн сложный. Стимулировать добровольных программистов наблюдать за тем как их друзья и члены семьи используют программу, побуждение их решать проблемы, которые возникают у других людей.

9. Мелочи остаются сломанными. Множество мелких деталей, которые улучшают интерфейс программы, не побуждают к работе над ними. Это такие детали как установка для окна наиболее подходящего размера и положения в момент его открытия, фокусировка и соответствующее управление по-умолчанию когда окно открывается, увеличение пользы от сообщений об ошибках и других текстов

или более точное отображение индикатором выполнения (progress bar) общего состояния работы. Поскольку эти вещи не побуждаю работу над ними часто проходят годы перед тем как они будут исправлены. Это оставляет у пользователей общее впечатление о плохом дизайне и может, в свою очередь, отбить желание участвовать у специалистов по юзабилити.

Решение: При планировании исправлений ошибок учитывайте сколько времени они потребуют. По возможности, планируйте мелкие изменения в интерфейсе на более раннее время если они могут быть сделаны быстро. Привлекайте дизайнеров интерфейса к этому планированию чтобы защититься от недостатков в юзабилити недооцененных потому что "это всего лишь проблема в интерфейсе".

10. Успокоение людей набором параметров. В любом программном проекте со множеством участников иногда кто-то будет против существования проблемам в дизайне. Когда участники работают по найму, то, обычно, они продолжают работать даже если не согласны с дизайном. Но с добровольцами, скорее всего, координатор проекта согласится успокоить участника добавлением параметра конфигурации для обсуждаемого поведения программы. Число, непонятность и простота таких параметров запутывает обычных пользователей, в то время как каждый оказывается наказан конечным распуханием программы и уменьшенной тщательности тестирования.

    Решение: Сильные координаторы проекта и культура простоты. Распределенное управление версиями программы тоже может помочь снять напряжение сделав проще для кого-то сопровождение своей собственной версии программы с желаемым поведением.

11. Пятнадцать пикселей славы. Когда доброволец добавляет новую функцию в популярное приложение можно понять его желание выделить это изменение чтобы иметь возможность показать что-то в интерфейсе и сказать "я это сделал". Иногда это приводит к новым пунктам меню для чего-то, что не должно было бы иметь дополнительного интерфейса. Наоборот, удаление запутывающих или бесполезных функций может вызвать гнев у программистов, которые их разработали.

    Решения: Предоставлять альтернативные средства, такие как блог, для предания известности участникам проекта. Ввести дизайнерский пересмотр изменений кода, которые затрагивают интерфейс

пользователя. Регулярно пересматривайте весь интерфейс спрашивая "Нам действительно нужна эта часть?".

12. Дизайн - "широкий", Сеть - "узкая". Программные проекты добровольцев обычно широко распределены. В них участвуют люди из разных городов, или даже разных континентов. Так что общение в рамках проекта по большей части осуществляется простым текстом, по электронной почте, мгновенными сообщениями, в чате или через систему отслеживания ошибок. Но разработка взаимодействия - задача многомерная, включающая размещение и поведение элементов в течение времени, а также организацию этих элементов везде в интерфейсе.

    Когда разработчики находятся в одной комнате, они могут обсудить схему взаимодействия используя доски, прототипы на бумаге, сказанные слова и движения. Но в Интернет эти средства часто не доступны, что значительно замедляет обсуждения и предрасполагает к недоразумениям.

    Решения: Разрабатывать и продвигать VoIP, видеочаты, виртуальные доски, программы для рисования и анимации, которые позволяют облегчить обмен дизайнерскими идеями через Интернет. И, где возможно, сохранить физические встречи разработчиков для совместной работы.

Наконец, несколько проблем специфичных для разработки Свободных программ.

13. Выпускать рано, выпускать часто, застревать. Общая практика "выпускать рано, выпускать часто" может стать причиной накопления элементов плохого дизайна. Когда предварительная версия ведет себя определенным образом и тестировщики использовали такое ее поведение, то они, естественно, будут возражать когда последующие предварительные версии ведут себя по другому. Даже если новое поведение лучше во всем. Это может отбить охоту у программистов вносить улучшения в интерфейс и привести к увеличению странных параметров конфигурации.

    Решение: Публиковать спецификации дизайна в максимально возможной ранней стадии процесса разработки чтобы тестировщики знали что им ждать в конце.

14. Посредственность через модульность. Хакеры Свободного ПО высоко ценят повторное использование кода. Часто они говорят о написании кода для выполнения определенного действия таким об-

разом чтобы другие программисты могли написать ”внешний интерфейс” (”front end”) (или множество альтернативных ”внешних интерфейсов”), который в действительности позволит людям его использовать. Они считают что важно иметь возможность заменить любой уровень системы на альтернативную реализацию.

Это хорошо для длительного существования системы, потому что позволяет избежать зависимости от любого из компонентов. Но это также ведет к отсутствию целостности, что ухудшает юзабилити. Особенно если интерфейсы между уровнями разрабатывались без учета юзабилити.

Например, большинство терминальных команд не предоставляют информацию о степени завершенности процесса или о том сколько времени она будет еще выполняться. Это традиционное поведение в терминале, но для ПО с графическим интерфейсом отображение прогресса выполнения важно. Если графическая программа разработана только как ”внешний интерфейс” к терминальной команде, то не может легко предоставить эту информацию.

Решение: Сначала разрабатывать пример графического интерфейса, чтобы требования этого интерфейса к нижним уровням были известны до того как эти уровни будут написаны.

15. Не связанные сообщества разработчиков. Когда вы что-либо делаете в компьютерной системе вы полагаетесь на программное обеспечение нескольких разных команд разработчиков. Например, если вы печатаете эту веб-страницу чтобы показать кому-то еще, это задание вовлечет в процесс не только веб-браузер, но также движок разметки страниц, оконный менеджер, инструментарий построения интерфейса, множество других библиотек, графическая подсистема, подсистема печати, драйвер принтера, файловая система, ядро, почти все из которых разработаны отдельными командами.

    Во многих случаях эти команды общаются с другими командами между собой достаточно редко. И, в отличие от своих собственнических соперников, почти все они имеют разные циклы разработки. Это делает реализацию улучшений юзабилити сложной и медленной, если эти улучшения требуют координации изменений между многими частями системы.

    Решения: Производители Свободного ПО могут координировать такие межкомпонентные функции если у них будут сотрудники, работающие на всех соответствующих уровнях стека программ. Также добровольные участники разработки различных программных

уровней могут встречаться на конференциях, организованных для этих целей.

Это длинный список проблем, но, я считаю, все они решаемы. В ближайшие месяцы я буду обсуждать примеры каждого из решений, чем лично помогу Свободному ПО добиться успеха.