http://gettingreal.37signals.com/GR_rus.php

Getting Real

Введение глава 1

Что такое Getting Real?

Хотите создать успешное веб-приложение? Тогда пришло время для подхода "Getting Real", легковесного, быстрого и в целом лучшего пути создания программного обеспечения.

- Getting Real -- это отказ от вещей, *представляющих* реальность (диаграммы, графики, схемы, стрелочки и модели) и *создание* реальной веши
- Getting Real -- это значит "меньше". Меньше массы, меньше программного обеспечения и его возможностей, меньше бумагомарания -- словом, меньше всего того, что является несущественным (а большая часть того, что, как вам кажется, критически важно, на самом деле таковым не является)
- Getting Real значит оставаться небольшим и шустрым.
- Getting Real начинает с интерфейса, с реальных экранов, которыми будут пользоваться ваши, клиенты. Это позволяет получить правильный интерфейс до того, как вы создадите неправильную программу.
- Getting Real -- это итерации и снижении стоимости изменений, Getting Real — это запуск и постоянное улучшение. То есть подход, идеальный для веб-приложений.
- Getting Real — это создание того, в чём нуждается клиент и исключение того, что ему не нужно.

Выгоды Getting Real

Getting Real дает лучшие результаты из-за того, что заставляет вас решать именно насущные проблемы, а не с вашими фантазиями на тему этих проблем. Другими словами, он заставляет вас иметь дело с реальностью. Getting Real отказывается от фунециональных спецификаций и подобных эфемерных документов в пользу реальных экранов. Функциональная спецификация -- это притворство, иллюзия договоренности, тогда как действительная веб-страница -- это реальность, то, что что будут видеть и использовать ваши клиенты. Только это и имеет значение и с помошью Getting Real вы достигнете этого гораздо быстрей, принимая решения на основе действительных вещей, а не абстрактных понятий. Наконец, Getting Real -- это подход, идеальный для веб-приложений. Дедовский способ поставки коробочных приложений вкупе с последующим двухгодичным ожиданием обновления уже изживает себя. В отличие от устанавливаемых у клиента приложений, их веб-аналоги могут развиваться и улучшаться каждый день. И Getting Real использует этот факт на полную катушку.

How To Write Vigorous Software

Предложение не должно содержать ненужных слов, абзац не должен содержать ненужных предложений. Точно так же рисунок не должен содержать ненужных линий и машина не должна содержать ненужных частей. Но это не значит, что писатель должен делать каждое предложение коротким или опускать все детали и лишь поверхностно описывать героев. Нет. Это значит, что каждое слово должно быть необходимым.

—*"The Elements of Style", Вильям Дж*

Больше никаких тяжёлых программ

Типичный подход: длительный бюрократический процесс, направленный в основном на прикрывание задниц. Типичный результат: раздутое, посредственное и быстрозабывающееся программное обеспечение.

Getting Real избавляется от…

- Временных диаграмм, простирающихся на месяцы или даже годы,
- Вымышленных функциональных спецификаций,
- Споров о масштабируемости
- Бесконечных собраний
- "Потребностей" нанимать десятки сотрудников
- Бессмысленных номеров версий
- Непорочных планов развития, предопределяющих идеальное будущее
- Бесконечных возможностей по настройкам
- Аутсорсинга поддержки
- Несоответствующего реальности тестирования
- Бесполезного бумагомарания
- Иерархии «сверху-вниз».

Вам не нужна куча денег, огромная команда или длительный цикл разработки для создания классного программного обеспечения. Всё это для медленных, неясных и неизменных приложений. У Getting Real другой подход.

Здесь вы узнаете...

- О том, насколько важно иметь философию
- Почему хорошо быть маленькими
- Как создавать меньше
- Как быстрее перейти от идеи к реальности
- Как набирать сотрудников
- Почему вы должны начинать проектировать начиная с интерфейса
- Почему навыки письма настолько важны
- Почему надо создавать меньше, чем ваши конкуренты
- Как продвигать и распространять ваше приложение
- Секреты успешной поддержки
- Как сохранить темп после запуска приложения,
- …и кое-что еще;)

Фокусируемся на крупных масштабах идеи. Мы не хотим нагружать вас кусками кода и CSS-трюками. Будем придерживаться главных идей и философии, которая управляет процессом Getting Real.

Эта книга написана для вас?

Вы предприниматель, дизайнер, программист или маркетолог, работающий на большой идеей.

Вы понимаете, что старые правила уже не нужны. Выпускаете свои программы на CD каждый год? Как 2002. Номенклатура версий? В мусорку. Вы должны создавать, начинать и достигать.

Или, может быть, вы еще не знакомы с быстрой разработкой и бизнес-структурами, но хотите узнать об этом.

Если вы подходите под эти описания, значит эта книга для вас.

Замечание: Хоть внимание и акцентируется на разработке веб-приложений, идеи применимы к действиям вне этой сферы. Предложения маленьких групп, быстром создании макетов, ожидание итераций и многие другие могут служить руководством, запускаете ли вы бизнес, пишете ли книгу, проектируете ли веб-сайт, записываете ли альбом или делаете что-то другое. Как только вы начнёте следовать принципам Getting Real в одной области жизни, вы поймёте, как это можно использовать в других областях.

О 37signals

Что мы делаем

37signals — небольшая команда, создающая простое и фокусированое программное обеспечение. Наши продукты помогут вам организовать совместную работу и быть более организованными. На данный момент, больше 350 тысяч пользователей используют наши веб-приложения для реализации своих целей. Наши приложения никогда вас не подведут.

Наш принцип работы

Мы уверены, программы слишком сложны. Так много возможностей, так много кнопок, столько времени, чтобы разобраться. Наши продукты сводят это к минимуму. Мы создаём продукты, которые работают шикарней, чувствуют пользователя лучше, позволяют вам вести разработку как вам удобно и весьма легки в использовании.

Наши продукты

На момент публикации этой книги мы имеем пять коммерческих продуктов и один OpenSource-фреймворк для веб-приложений.

Basecamp управляет проектами. Вместо диаграмм Гантта, причудливых графиков и тяжёлых таблиц статистики Basecamp предоставляет форумы для общения, todo-листы, простое планирование, совместную работу. Пока многие согласны с тем, что это—лучший путь.

Campfire предоставляет просто групповой чат для бизнеса. Компании понимают, как важен может быть постоянный чат в реальном времени для рабочей группы. Обычные чаты являются слишком большими для быстрого обмена сообщениями наедине, но их не хватает для троих или более человек. Campfire решает эту проблему и многие другие.

Backpack — альтернатива для тех, кому надо управлять своей жизнью. «Организация вашей жизни в 25-ти простых шагов». Персональный управляющий вашей информацией. Backpack — это просто страницы, заметки, задачи, напоминания по телефону и электронной почте. Это инновация в категории продуктов, которая страдает от status-quo-itis. Томас Вебер из Wall Street Journal назвал это лучшим продуктом в своём классе и Дэвид Погу из New York Times назвал это «очень крутым» организационным инструментом.

Writeboard позволяет вам писать и совместно использовать тексты. Это альтернатива «раздутым» текстовым процессорам, которые убивают около 95-ти процентов того, что вы пишете.

Ta-da List сохраняет списки задач и организует вас онлайн. Создайте списки для себя или совместно используйте их с другими. Нет способа проще, чтобы добиться своей цели. На данный момент создано более ста тысяч списков 3f;очти с миллионом задач.

Ruby on Rails, для разработчиков то, что надо. Открытый фрейморк на Ruby для быстрого и лёгкого написания реальных приложений. Rails занимается всей рутиной в то время, как вы сосредоточены на своей идее.

Протесты и оговорки

Некоторые ответы на ваши письма:

«У меня это не работает»

Getting Real — это система, которая работала именно для нас. То есть, идеи этой книги не применимы к каждому проекту на этой земле. Если вы разрабатываете оружейные системы, системы управления ядерными разработками, банковские системы, или другие жизненно и финансово критичные системы, вам надо хорошенько подумать, перед тем, как использовать эту методику.

И не надо выбирать: всё или ничего, даже если вы не можете использовать Getting Real полностью, здесь есть по крайней мере несколько идей, полезных для вас.

«Эту идею изобрели не вы»

Мы и не утверждаем, что изобрели эти методы. Многое из этого уже было в течении долгого времени, приобретая разные формы. Не раздражайтесь, если вы нашли нечто такое, что уже читали на каком-то блоге или в книге, изданной лет 20 назад. Определённо, это возможно. Эти методы не принадлежат 37signals. Мы просто рассказываем вам, как мы работаем и что нам помогло.

«Вы всё видите в чёрно-белых цветах»

Потерпите нас. Мы считаем, лучше представить идеи резко, чем тихо шептать об этом на ухо. Если это кажется дерзким и высокомерным, пусть так и будет. Мы предпочитаем быть провокационными чем постоянно ныть «может быть, если…» Конечно, будут времена, когда эти правила должны будут измениться или отмениться. И некоторые из этих тактик, возможно, не подходят к вашей ситуации. У вас есть своя голова на плечах, решайте сами.

«В моей компании это не будет работать»

Наверное вы слишком большие для Getting Real? Даже Microsoft использует Getting Real, сомневаемся, что ваша компания больше них.

Даже если ваша компания всё ещё создаёт долгосрочные цели с большими командами, способы достигнуть Getting Real всё ещё есть. Только надо это делать по-тихоньку. Когда слишком много вовлечённых людей, ничего не получится. Чем вас меньше, тем быстрее и лучше можно сделать всё.

Естественно, это может потребовать некоторых продаж. Предоставьте Getting Real вашей компании. Покажите им эту книгу. Покажите им реальные результаты, которых вы можете достигнуть быстрее и с меньшей командой.

Объясните, что Getting Real — ничтожные риски, требует мало инвестиций, чтобы проверить его в деле. Посмотрите, сможете ли вы отделиться от базовых понятий, чтобы доказать возможности Getting Real на маленьком проекте. Продемонстрируйте результаты.

Или, если вы действительно хотите быть напористым, идите на хитрость. Сделайте всё тихо и продемонстрируйте реальные результаты. Этот подход использовала команда Start.com, используя Getting Real в Microsoft.

Свободное плавание Start.com (проект Microsoft):

В больших компаниях процессы и встречи — норма. Многомесячное планирование и утверждение деталей с целью постоянного согласия в том, что является «правильным» для клиента.

Это может быть правильным подходом для коробочного ПО, но в интернете мы имеем невероятное преимущество. Просто попробуйте это! Позвольте пользователю говорить вам, правильно это или нет. Вы можете исправить и обновить это хоть сразу! Нет ничего важнее слов клиента — сопротивление убеждению участвовать на долгих встречах.

Легче сказать, чем сделать:

Месяцы планирования не так нужны. Месяцы написания спецификаций не так важны — спецификации должны иметь детали, выясненные в течении всей фазы развития. Не пробуйте закрыть все открытые проблемы и сохранить каждую деталь до начала разработки.

Сделайте меньше возможностей, но пусть они будут качественными. Вы не нуждаетесь в большом взрыве с новым релизом и кучей возможностей. Дайте пользователю немного, чтобы он мог понять и привыкнуть.

Если есть незначительные ошибки, завершите основную цель, а потом постепенно исправляйте ошибки. Так вы быстрее сделаете обратную связь с пользователем лучше. Идеи могут великолепно выглядеть на бумаге, но на практике оказываются не такими уж и оптимальными. Чем скорее вы осознаете фундаментальные проблемы вашей идеи, тем лучше.

Как только вы начнёте всё делать быстро и реагировать на обратную связь с клиентом, вы установите прочную связь с клиентом. Помните, что цель состоит именно в том, чтобы получить постоянного клиента, создавая то, что они хотят.

— *Саназ Ахари, менеджер Start.com, Microsoft.*

Начало глава 2

Делайте меньше

Меньше соревнуйтесь

Народная мудрость гласит: чтобы подавить своих конкурентов, вы должны переиграть их. Если у них четыре возможности, у вас их должно быть пять, пятнадцать, двадцать… Если они тратят N, то вы должны потратить NN. Если у них есть 20, то у вас должно быть 30.

Это — тупик. Дорогой и параноидальный путь создания продуктов. Защищающиеся, параноидальные компании не могут предположить, что остаются позади. Они не лидеры, а последователи.

Если вы хотите сделать свою компанию ведомой, а не ведущей, вы можете прекратить читать эту книгу прямо сейчас.

Так что же делать тогда? Ответ: меньше. Сделайте меньше чем ваши конкуренты, чтобы ударить по ним. Решайте простые проблемы и оставьте сложные, противные проблемы всем

остальным. Вместо превосходства попробуйте делать недостаточно.

Мы будем раскрывать понятие «меньше» в этой книге, но для начинающих это:

- Меньше возможностей
- Меньше опций и настроек
- Меньше структура компании
- Меньше встреч и абстракций
- Меньше обещаний

В чём ваша проблема?

Разрабатывайте ПО для себя

При создании программ большую часть времени будет занимать решение ваших собственных проблем. Вы и будете тем самым потенциальным клиентом, соответственно и знание о необходимом у вас уже есть. Это даёт вам отличное начало для последующего распространения продукта.

Главное здесь — понять, что вы не одни. Если у вас есть проблема, вероятно сотни тысяч других людей находятся в том же положении. Это ваш рынок. Это ведь не сложно?

Basecamp был создан при решении проблемы: как студия дизайна, мы нуждались в простом способе общения с нашими клиентами о проектах. Мы начали это делать через экстранет, который мы обновляли вручную. Но изменение HTML каждый раз при обновлении проекта — не самое приятное занятие. Эти проектные сайты всегда казались устаревшими и, в конечном счёте, были заброшены. Это расстроило нас, потому что оставили нашу работу неорганизованной и без общения с клиентами.

Таким образом, мы пришли к рассмотрению других вариантов. Каждый продукт либо не предоставлял возможностей, в которых мы нуждались либо был сложен, имел кучу возможностей, которые не представляли для нас интереса: составление счетов, жёсткое управление доступом, диаграммы, графики… Мы знали, что есть лучший путь. Итак, мы решили создать собственное приложение.

Если вы решаете собственную проблему, вы создаёте инструмент с душой. И это является ключевым. Это означает, что вы сделаете всё отлично. И это является лучшим вариантом.

Финансируйте себя сами

Деньги извне — план »B«

Целью многих стартапов является привлечение денежных средств инвесторов. Но помните, если вы принимаете помощь инвесторов, вы несёте ответственность перед ними. Ожидания от стартапа вырастут. Инвесторы довольно быстро потребуют свои деньги обратно. Они (деньги) будут превыше качества разрабатываемого продукта.

Сейчас, чтобы начать, не требуется многого. Аппаратные средства дешевы и большое количество ПО бесплатно и имеет открытые исходные коды. И страсть не приходит с ценником.

Так что сделайте всё, что вы можете с теми деньгами, что уже у вас есть. Усиленно думайте и расставьте приоритеты. Что вы можете сделать с тремя людьми вместо десяти? Что вы сделаете с 20 тысячами долларов вместо 100 тысяч? Что вы сделаете за три месяца вместо шести? Что, если вы продолжите выполнять вашу ежедневную работу и одновременно будете создавать ваше приложение на стороне?

Ограничения заставляют думать творчески

Имея ограниченные ресурсы, вы будете вынуждены считаться с ограничениями на ранней стадии и сильнее это ощущать. И это хорошо. Ограничения заставляют делать по-новому.

Ограничения также вынуждают развивать идею как можно скорее — ещё одна хорошая вещь. Месяц или два обдумывайте идею и поймите, стоит ли она того. Если да и вы будете быстры и жизнестойки, вам не понадобятся инвесторы. Если ваша идея еще «не созрела», вам придётся вернуться к чертежам. По крайней мере, теперь вы знаете какие будут препятствия, не тратя на это многие месяцы и годы. И можете легко отступить.

Если вы будете создавать ПО лишь ради быстрых денег — увидите, что быстрая прибыл маловероятна. Так что сосредоточьтесь на создании качественного инструмента, с которым ваши клиенты захотят работать в течении долгого времени.

Два пути

Джейк Уолкер создал одну компанию при помощи инвестора (Disclive) и другую — без его помощи (The Show). Она обсуждает различия между этими способами.

Деньги не были источником всех проблем, но проблемы приходили вместе с ними. Ожидания были выше. Люди работали ради зарплаты, хотелось создать и продать бизнес, либо быстрее вернуть деньги инвесторам. В случае с первой компанией нам пришлось действовать больше.

С The Show мы поняли, что можем создать намного лучший продукт с меньшими затратами. Только на это понадобится больше времени. Мы рисковали небольшим количеством собственных денег и ожидали, что люди на энтузиазме будут стремиться к качеству и скорости работы. И компания осталась (и, вероятно, продолжит быть) с небольшим оборотом. Начиная с первого проекта мы полностью были на самообеспечении. Из-за маленьких сроков работы с клиентами у нас вообще не было необходимости вкладывать действительно немаленькие средства в бизнес. И мы хотели не вырастить бизнес и продать его, а развивать его и продолжать извлекать из этого прибыль.

— *Комментарий на блоге Signals vs. Noise.*

Фиксируйте время и бюджет, делайте возможности гибче

Запускайте вовремя и согласно смете

Лёгкий способ начать вовремя и уложиться в бюджет: фиксируйте время и бюджет. Никогда не отдавайте больше времени или денег проблеме, умерьте пыл.

Бытует миф: мы можем начать вовремя, уложившись в бюджет и реализуя всё предполагаемое. Это практически никогда не выходит, но когда так происходит, от этого страдает качество.

Если вы не укладываетесь в отведённые время и бюджет, не увеличивайте их. Вместо этого сократите возможности. Время добавить их чуть позже будет всегда.

В начале лучше будет меньше запланированных возможностей, чем посредственным, громоздким и с кучей дыр безопасности. Оставьте волшебство Копперфильду. У вас реальный бизнес и реальный продукт.

Вот выгоды фиксированных времени и бюджета, и гибких возможностей.

Расставление приоритетов

Выясните, что действительно важно. Что сподвигло на создание продукта? Это вызовет ограничения, которые сподвигнут вас на принятие быстрого, точного и жёсткого решения вместо тупого «мямляния».

Действительность

Фиксация ожиданий — ключевой момент. Если вы фиксируете время, бюджет и возможности, вы не поставите их выше качества. Несомненно, вы можете поставить кое-что, но что именно?

Гибкость

Способность изменяться является ключевой. Жёсткие рамки не позволят изменяться. Добавление гибких возможностей приведёт к альтернативам, основанным на вашем опыте разработке продуктов. Гибкость — ваш друг.

Рекомендуем сократить возможности. Лучше сделать половину продукта, но качественно, чем недоделку.

Раз, два, три...

Возможно ли выполнить проект за год до намеченной даты? Когда-нибудь, наверное.

—Фред Брукс, инженер ПО и программист.

Заведите себе врага

Боритесь

Иногда, лучший способ узнать, каким должно быть ваше приложение — это узнать, каким оно не должно быть. Пусть это будет врагом вашего приложения и вы будете видеть свет, на который вы должны идти.

Когда мы решили создать систему управления проектами, мы знали, что MS Project был гориллой в комнате. Боясь горилл, мы использовали это для мотивации. Мы решили, что Basecamp будет ему полной противоположностью, анти-Project.

Мы поняли, что управление проектом — это не диаграммы и графики, отчёты и статистика; это общение. Это также не менеджер проектов, сидящий высоко и раздающий проектные планы. А тот, кто несёт ответственность и делает проектную работу.

Нашими врагами были Диктаторы Управления Проектами, которые имели обыкновение

«бить кнутом». Мы хотели демократизировать управление проектом, сделать так, чтобы каждый был его частью (включая клиента). Проекты становятся лучше, когда у каждого есть интересы в нём.

Когда начали работу над Writeboard, мы знали, что есть конкуренты этому продукту с гораздо большим количеством возможностей. Таким образом мы решили выделить «простоту» вместо этого. Мы создали приложение, которое позволило людям легко делиться идеями и сотрудничать, без увязания в несущественных особенностях. Если что-то было несущественным, мы отказывались от этого. А через три месяца после запуска было создано более чем 100 тыс. Writeboard-ов.

Когда мы начали работу над Backpack, нашими врагами были структура и твёрдые правила. Люди должны организовывать информацию как им удобно, не основываясь на всеобщих убеждениях «как надо».

Бонус, который вы получаете при наличии врага — весьма ясное маркетинговое сообщение. Людей топит конфликт. И они также сравнивают продукт с другими продуктами. Выбирая врага, вы даёте людям то, что они хотят увидеть. И они не только будут понимать ваш продукт лучше и быстрее, они встанут на вашу сторону. Это безошибочный способ получить внимание и зажечь страсть.

Исходя из всего сказанного — важно быть не слишком одержимым борьбой с врагом. Тщательно анализируйте другие продукты и вы будете ставить ограничения своим идеям. Наблюдайте, и только потом двигайтесь к собственному видению и к собственным идеям.

Не следуйте за лидером

Специалисты по маркетингу (и все люди) хорошо «обучаются» следовать за лидером. Естественный исинтинкт должен выяснить, что работает для борьбы, а затем пробуйти превзойти это — чтобы быть дешевле своего конкурента, который конкурирует ценой, или быстрее, если он конкурирует скоростью. Проблема в том, что как только потребитель купил чью-либо неверную историю и верит этой лжи, убеждать его переключиться — то же самое, что убеждение его в том, что он был неправ. А люди очень не любят признавать свою неправоту.

Вместо это расскажите другую историю и убедите его, что это ваша история важнее, чем история, которой он сейчас верит. Если ваше соревнование в том, кто быстрее — вы должны стать более дешёвым. Если кто-то продаёт историю здоровья, вы должны продать историю удобства. Не только говоря «мы более дешёвые», но и предоставляя реальную историю, отличающуюся от существующих.

— Сет Годин, автор/предприниматель, Be a Better Liar.

В чём ключевая проблема?

Один из быстрейших способов сделать из самого себя проблему состоит в том, чтобы смотреть на то, что делают ваши конкуренты. Это особенно подходило для нас в BlinkList. Так как мы начали приблизительно 10 других соц. сетей услуг хранения веб-закладок. Некоторые люди даже начали делать крупноформатные таблицы с детальными сравнениями особенностей.

Особенно, это может быстро сбить с пути. Вместо этого мы остаемся сосредоточенными на большой картине и нас продолжают спрашивать, что является ключевой проблемой, которую мы пробуем решить и как решаем это.

— Майкл Рейнинг, соучредитель MindValley & Blinklist.

Никакой рутины

Ваша страсть или её нехватка — будет ясно видна

Чем меньше в вашем приложении рутины — тем лучше.

Если рутинной работы немного и она управляема, вы можете наслаждаться

Если ваше приложение не волнует — это плохо. Если вы делаете это только из-за денег — это проявится. И если вы обожаете свое приложение — это проникнет в конечный продукт. Люди умеют читать между строк.

Присутствие страсти

В проекте, где смысл часто спорный, субъективный или непостижимый, немногое более очевидно и ясно, чем присутствие страсти. Может дизайн продукта вызывает у вас восхищение или обдаст холодом; в любом случае трудно не обнаружить, что его делали с душой.

Энтузиазм проявляется с готовностью, конечно, но безразличие — одинаково несмываемо. Если ваши взгляды не охватывают подлинную страсть к работе, это становится пустотой, которую почти невозможно скрыть, независимо от того, как продуманно или привлекательно всё было разработано.

— *Кой-Вин, Subtraction.com и соучредитель Behavior LCC.*

Пекарня

Американский бизнес призывает к развитию идеи, к получению из этого выгоды, к последующей продаже, в то время как это становиться выгодным, а затем собирает доходы или диверсифицирует развитие. Это высасывает все. Моя идея: Любите печь, продавать ваш хлеб, который нравится людям, продавайте больше. Держите пекарню в движении, потому что вы делаете хорошую пищу и людей счастливыми.

— *Ян Маккэй, член Fugazi и совладелец Dischord Records.*

Оставайтесь небольшими глава 3

Меньше размеры

Чем вы беднее, тем проще это изменить

Чем больше объект, тем больше нужно энергии для его изменения. Это также верно в деловом мире, как и в обычном.

В вебе изменения должны проходить легко и дешево. Если вы не можете изменяться налету, вы проиграете тому, кто может. Поэтому вы должны стремиться к меньшим размерам.

Размеры увеличиваются, если

- Есть долгосрочные контакты

- Большой штат сотрудников
- Постоянные решения
- Совещания о других совещаниях
- Толстый процесс
- Инвентарь (физический или умственный)
- Аппаратное и программное обеспечение, технологии закрыты.
- Проприетарные форматы данных
- Прошлое управляет будущим
- Долгосрочные цели
- Офисная политика

Размеры уменьшаются, если

- Своевременные размышления
- Мультизадачность членов команды
- Охват ограничений (не пробуя их снять)
- Меньше программного обеспечения, меньше кода
- Меньше возможностей
- Небольшая команда
- Простота
- Открытые исходные коды
- Открытые форматы данных
- Открытая культура, которая облегчит совершение ошибок

Меньшие размеры позволят вам быстрее изменять необходимое. Вы можете реагировать и развиваться. Вы можете сфокусироваться на хороших идеях и выкинуть плохие. Вы можете слушать своих клиентов и отвечать им. Вы можете интегрировать новые технологии сейчас, а не потом. Вместо авианосца вы управляете лодкой.

Например, давайте представим одну компанию, у которой есть несколько продуктов с небольшим количеством особенностей. А на другой стороне — компания с большим числом программ и с большим числом особенностей. Появляется новая технология, как AJAX или новое понятие как теги. Кто из них в состоянии быстрее приспособить эти технологии к своему продукту? Команда с большим количеством программного обеспечения и большим количеством особенностей и с годовым планом или команда с меньшим количеством программного обеспечения и органическим: «давайте сосредотачиваться на процессе»?

Очевидно, что небольшая компания находится в лучшем положении и лучше приспосабливается к реальным требованиям рынка. Более крупная компания будет вероятно все ещё обсуждать изменения или проталкивать их через бюрократию. Небольшая компания будет на два шага впереди, в то время как большая компания все ещё думает куда идти.

Ловкие, проворные и менее массовые компании могут быстро менять свою бизнес-модель, продукт, особенности и маркетинговое сообщение. Они могут менять приоритеты, объединять продукты и что самое важное у них есть право передумать.

Снизьте стоимость перемен

Будьте гибкими, снижая препятствия для перемен

Перемена — ваш лучший друг. И если ваши конкуренты могут измениться быстрее, чем

вы, у вас большие проблемы. Если перемены даются слишком дорого — вы мертвы.

Вот где бедность действительно поможет вам. Способность к переменам с мелкой монетой — та вещь, которая есть у маленьких команд по умолчанию, у больших команд этого не будет никогда. Это то, когда большие ребята действительно завидуют маленьким ребятам. В большой организации на какое-то изменения потребуется неделя, а в малой — это займет всего один день. Это преимущество бесценно. Дешевые и быстрые перемены — это секретное оружие малого бизнеса.

И помните: Вся наличность, весь маркетинг, все люди в мире не могут купить проворство, которое вы уже получаете оттого, что вы маленький.

Непредсказуемость

Непредсказуемость одно из основополагающих принципов проворства, и очень близко к чистому волшебству. Неожиданно появляющиеся свойства не проектируются или строятся, они просто случаются, как динамичный результат остальной части системы.

«Непредсказуемость» происходит с середины 17-го столетия из латинского словосочетания «непредвиденный случай». Вы не можете спланировать это, но вы можете культивировать окружающую среду, в которой позволите этому случиться и извлечь из этого выгоду.

Классический пример непредсказуемости лежит в поведении птиц, которые держатся вместе. Компьютерное моделирование может использовать только лишь три простых правила (например, параллельные линии не пересекают друг друга), а внезапно вы получаете целый комплекс поведения, так как пушинка двигается и ведет свой путь грациозно через небо, изменяя вокруг себя препятствия. Ни одно из этих правил не предполагает такого поведения (как, например, преобразование такой же формы вокруг препятствия), это не указано в установленных правилах; это появляется из динамики системы.

Простые правила, как с моделированием птиц, приводят к сложному поведению. Комплекс правил, как с налоговым законодательством в большинстве стран, приводят к глупому поведению.

Методы разработок стандартного программного обеспечения имеют неудачный «обратный эффект» — удаление любой возможности непредсказуемо меняет поведение программы. Большинство попыток оптимизации, обязывают очень явно сокращать широту и контекст взаимодействий и взаимоотношений, которые являются источником непредсказуемости. Пример птиц, летящих вместе, это хорошая система, в которой взаимодействия и взаимоотношения создают интересное поведение, и одно и тоже, сколько бы птиц не исключили из системы.

Разрабатываете ли вы требования перед тем, как приступаете к реализации; оптимизируете ли код; или изобретаете сложную навигацию и сценарии пользователей, результат один С8; тот же: чрезмерно сложная и глупая система, вместо изящной и легкой.

Будьте небольшими. Будьте проще. Позвольте вмешаться динамике и непредсказуемости.

— *Эндрю Хант, The Pragmatic Programmers*

Три Мушкетера

Используйте команду из трех человек для версии 1.0

Для первой версии вашего приложения, можно начать только с тремя людьми. Это магическая цифра, которая обеспечит вам достаточный человеческий ресурс, а еще позволит

оставаться быстрым и проворным. Начните с разработчиком, проектировщиком, и чистильщиком (sweeper), — тот, кто будет между ними.

Сейчас, это безусловно вызов, в том, чтобы построить приложение только с несколькими людьми. Но если вы имеете правильную команду, то это того стоит. Талантливым людям не нужны бесконечные ресурсы. У них есть энтузиазм. Их творческий потенциал можно использовать для решения проблем. Отсутствие человеческого ресурса означает, что вы будете иметь дело с выборами из альтернативных вариантов раньше и уже в процессе — и это правильно. Это выявит ваши приоритеты. И вы раньше будете связаны одной петлей с вашими людьми.

Если вы не можете построить версию 1.0 с тремя людьми, то или вам нужны другие люди, или необходимо понизить начальные условия. Помните — хорошо держать вашу первую версию маленькой и компактной. Вы быстро увидите, если ваша идея хороша — у вас будет чистая и простая основа для последующей надстройки.

Metcalfe's Law и проектные команды

Удерживайте команду небольшой как только это возможно. Metcalfe's Law гласит: «система коммуникации растет прямо пропорционально квадрату количества потребителей системы». А это заключение относится непосредственно к проектным командам: «Эффективность команды — прямо пропорционально квадрату числа членов в команде». Я начинаю думать, что три человека — это оптимально для 1.0 редакции продукта… Начните, сокращая количество людей, которых вы планируете добавить в команду, а затем сокращайте еще.

— Marc Hedlund, entrepreneur-in-residence at O'Reilly Media

Коммуникационный поток

Потоки коммуникаций просты в маленьких командах. Если вы — единственный человек в проекте, коммуникация проста. Единственный путь коммуникации — это вы и клиент. Если число людей в проекте увеличивается, это не увеличивает на столько же число коммуникаций между вами и клиентами. Число коммуникаций не возрастает аддитивно, как увеличение людей, поток увеличивается независимо и пропорционально квадрату числа людей.

— Steve McConnell, Chief Software Engineer at Construx Software Builders Inc. (from Less is More: Jumpstarting Productivity with Small Teams)

Принимайте ограничения

Пусть ограничения ведут вас к творческим решениям.

Многого недостаточно, чтобы не идти кругами. Мало времени. Недостаточно денег. Мало людей.

Это хорошо.

Вместо рассуждений об этих ограничениях, примите их. Пусть они ведут вас. Ограничения управляют новизной и центром силы. Вместо попытки сместить их, используйте как преимущество.

Когда в 37signals строили Basecamp, у нас было много ограничений:

- Мы начинали как дизайнерская фирма
- Существовала текущая клиентская работа

- Было 7 часов разницы во времени (Давид делал программирование в Дании, а остальные находились в Штатах)
- Маленькая команда
- Не было внешнего финансирования

Мы приняли ограничения и взяли большие задачи, разделили их на маленькие куски, чтобы попытаться выполнить их по одному. Мы двигались шаг за шагом и с определенными приоритетами.

Ограничения вынудили нас приходить к творческим решениям. Мы понизили стоимость наших изменений, всегда делая меньшее программное обеспечение. Мы предоставили людям достаточно возможностей, только для того чтобы разрешить их собственные проблемы, а не указать им собственный путь. Разница во времени и расстояние между нами стало серьезным фактором в нашей коммуникации. Вместо встречи, мы связывались почти исключительно через мессенджеры и электронную почту, которые вынудили нас добраться до цели еще быстрее.

Ограничения — часто замаскированы. Забудьте о вкладе капиталов с риском, длинном цикле релизов, и быстром найме людей. Вместо этого, работайте с тем, что у вас есть.

Будьте собой

Дифференцируйте себя от больших компаний, будьте дружественнее и доступнее

Много маленьких компаний делают ошибку в попытках действовать как большие. Как будто бы они не замечают свой размер. Это как слабость, комплекс, который нужно скрывать, и это слишком плохо. Быть маленьким фактически обладать огромным преимуществом, особенно, когда это касается коммуникации.

Маленькие компании наслаждаются меньшими формальностями, меньшей бюрократией, и большей свободой. Маленькие компании ближе к клиенту по умолчанию. Это означает, что они могут связаться прямо и лично с клиентами. Ваш сайт и ваш продукт могут иметь человеческие тексты, вместо зондирования корпоративных трутней. Ваши клиенты будут говорить непосредственно с вами, а не с продавцами, которые тянут вниз компанию.

Есть также преимущества во внутренних коммуникациях. Нет никакой потребности в куче формальностей и многоразовых отчетах на все. Каждый в процессе может говорить открыто и честно. Это снимает оковы с потока идей — одно из самых больших преимуществ небольшой компании.

Будьте с гордостью, вызывающе правдивы

Вы, возможно, думаете, что клиента могут дурачить преувеличения числа штатных сотрудников в вашей компании или широта ваших предложений. Те, клиенты, которых вы действительно хотите, будут всегда искать правду — через интуицию или вычисления. Потом останется только смущение морально неоправданной лжи в прошлом, и ни одна из тех ситуаций не приведет к тому, что в бизнесе имеет значение: к взаимовыгодным отношениям с людьми, с теми, кто имел реальную потребность в предложенных услугах. Лучший курс — быть с гордостью, вызывающе правдивым и сообщать о точном размере компании и широте предложений.

— Khoi Vinh, Subtraction.com and co-founder of Behavior LLC

Всегда в любое время

Неважно, в каком бизнесе вы находитесь, хорошее обслуживание клиентов должно быть. Мы требуем этого в услугах, мы используем это, почему бы нам ни подумать, что наши клиенты ждут того же от нас?

С самого начала мы сделали легким и прозрачным все, чтобы клиенты могли связаться с нами по любому вопросу. На нашем сайте мы указали бесплатный номер и наши мобильные телефоны и на наших визитках каждый из нас оставляет контактную информацию. Мы делаем упор на то, что клиенты, могут добраться до нас и связаться в любое время, и неважно в чем, возможно, была бы проблема. Наши клиенты ценят этот уровень доверия, и никто никогда не злоупотребил нашим обслуживанием.

— Edward Knittel, Director of Sales and Marketing, KennelSource

---

Приоритеты глава 4

В чем идея

Явно и точечно определите видение для вашего приложения.

Что ваше приложение должно делать? Это действительно все?

Перед тем как вы начнете проектировать или кодировать что-либо, вам нужно знать цель вашего продукта — видение. Думайте. Почему это может существовать? В чем будут отличия от других подобных программ?

Это видение будет вести вас, и ваши решения держать на последовательном пути. Каждый раз, когда есть сомнения в решении, можно спросить себя: «Мы остаемся верными видению?»

Ваше видение должно быть кратким и содержать идею. Вот — виденье для каждого нашего продукта:

- **Basecamp**: Управление проектом — это коммуникация
- **Backpack**: Соединить детали вместе
- **Campfire**: Чат группы против IM
- **Ta-da List**: Конкуренция post-it запискам
- **Writeboard**: Слово — массовое оружие (Word is overkill)

С Basecamp, например, виденье было таким: «Управление проектом — это коммуникация». Мы считаем, что эффективная коммуникация втягивает по инерции собственные инвестиции и силы в проект. Каждый получает работу в направлении общей цели. Мы знали, что Basecamp достигнет этого, все другие линии провалились бы.

Это виденье заставляло нас делать так, чтобы Basecamp был как можно более открытым и прозрачным. Вместо ограничений коммуникации в пределах фирмы, мы предоставили такой же доступ и клиентам. Мы меньше думали о разрешениях и больше о поощрении, чтобы все приняли участие в проекте — это то, почему мы опустили диаграммы, графики, статистику и электронные таблицы. Вместо чего сосредоточились на приоритетах коммуникаций: сообщениях, комментариях, списках и распределению файлов.

Найдите свою большую идею и примите решение о видении, все маленькие решения

в будущем станут проще и легче.

Философия Whiteboard

Andy Hunt и я делали debit card transaction switch. Главным требованием было, что потребитель дебетовой карты не должен иметь одну и ту же сделку, совершенную дважды. Другими словами, такая проблема считалась бы ошибкой и обрабатывалась бы только одна сделка.

Так что, мы написали об этом на нашем общем whiteboard: «Ошибка на пользу потребителей».

Это присоединило к себе дюжину других принципов. Совместно, они ведут все хитрые решения, которые происходят, когда строишь что-нибудь комплексное. Вместе эти принципы создают внутреннюю и внешнюю последовательность действий.

— *Dave Thomas, The Pragmatic Programmers*

Создавайте молитвы

Организациям нужны указательные столбы. Им нужен план; работникам каждый день нужно знать, когда они просыпаются, почему они собираются идти на работу. Этот план должен быть кратким и сладким, и затрагивать все: Почему вы существуете? Как это мотивируете? Я называю это молитвой — описание в трех-четырех словах причин, по которым вы существуете.

— *Guy Kawasaki, author (from Make Mantra)*

Пренебрегайте деталями в начале

Работайте от большего к меньшему

Мы сумасшедшие до деталей.

- Пространство между объектами
- Совершенный цвет
- Совершенные слова
- Четыре линии кода вместо семи
- 90% vs 89%
- 760px vs 750px
- $39/month против $49/month

Успех и удовлетворение находится в деталях.

Однако, успех не единственная вещь, которую вы найдете в деталях. Вы также найдете — застой, разногласие, встречи, и задержки. Эти вещи могут убить моральное состояние и снизить вероятность успеха.

Как часто вы сидите над одной строчкой кода в течение целого дня? Как часто ваша работа сделанная за один день не дала никакого прогресса? Это случается, когда вы сосредоточиваетесь на деталях, слишком рано. У взыскательного человека будет еще много времени на детали. Просто отложите это.

Не волнуйтесь о размере шрифта в заголовках. Вы не нужна совершенная тень. Вам не нужно перемещать кнопку на три пикселя вправо или влево. Просто поместите материал

на страницу. А затем используйте. Убедитесь, что это работает. Позже вы можете все усовершенствовать.

Детали проявляются, пока вы используете то, что вы строите. Вы будете видеть, чему нужно уделить больше внимания. Вы будете знать, какие выбоины надо замостить, потому что вы будете продолжать биться об них. Именно тогда, на них следует обратить внимание, не раньше.

Дьявол в деталях

Если вы начинаете втягиваться в детали немедленно, можете быть уверены что рисунок будет плохим. Фактически, вы целиком не понимаете суть дела.

Вы должны получить пропорции для целого. А затем делать эскиз наибольших объектов, переходя к самым маленьким. Эскиз должен быть простым вплоть до этого пункта. Затем вы можете возобновить штриховку, которая приведет объем в чувство.

Вы начинаете только с трех тонов (светлый, средний, темный). Это будет тональный эскиз. Затем для каждой части вашего рисунка оцените три тональных тени и примените их. Сделайте это, пока объемов нет (требует многоразового повторения)…

Работайте от большего к меньшему. Всегда.

— Patrick Lafleur, Creation Objet Inc. (from Signal vs. Noise)

Проблема тогда, когда это проблема

Не тратьте бесцельно время на проблемы, которых у вас еще нет

Вам действительно нужно волноваться о вычислениях для 100 000 потребителей сегодня, если это будет у вас через два года?

Действительно вам нужно нанять восемь программистов, если сегодня нужно только три?

Действительно сейчас нужны 12 первоклассных серверов, если вы можете обойтись двумя на протяжении года?

Люди часто тратят слишком много времени на попытки решить проблемы, которых у них нет. Не делайте этого. Мы начали делать Basecamp без клиентов! С тех пор, как мы взялись за разработку продукта, у нас было 30-дней. Мы использовали это время, чтобы разрешить более срочные проблемы, а затем, после создания основы, мы попытались провести подсчеты и определить, сколько же клиентов у нас будет.

Это было простым C8; отличным решением, без ненужных смет и усилий.

Не работайте над материалом, пока вы фактически не должны этого делать.
Не надстраивайте. Увеличивайте технические средств и системное программное обеспечение по мере необходимости. Если вы задержитесь на неделю или две — это не конец света. Просто будьте честным: объясните вашим клиентам, что вы растете и решаете некоторые проблемы из-за этого. Они, возможно, не будут в восторге, но оценят искренность.

Совет: Принимайте решения вовремя, когда у вас есть доступ к реальной информации, которая необходима. Тем временем, вы сможете сосредоточить внимание на вещах, которые требуют непосредственной заботы.

Работайте с правильными клиентами

Найдите основной рынок для вашего приложения и сосредоточьтесь исключительно на нем

Клиент не всегда прав. Правда в том, что вам придется определять кто прав, а кто ошибается — в рамках вашего приложения. Хорошая новость в том, что интернет делает поиск правильных людей легче, чем когда-либо.

Если вы ориентируетесь на всех, вы не удовлетворите никого

Когда мы построили Basecamp, мы сконцентрировали свой маркетинг на дизайн студиях. Сузив рынок, мы увеличили вероятность привлечения страстных клиентов, которые, в свою очередь, проповедовали продукт как Евангелие (evangelize the product). Знайте, для кого ваше приложение действительно предназначается, и сконцентрируйтесь на их удовлетворении.

Лучший вызов, который когда-либо мы сделали

Решение нацелить Campaign Monitor строго на рынок сетевого проектирования был лучшим вызовом, который мы когда-либо сделали. Это позволило нам легко выделить особенности, которые были действительно полезны и важны. Мы привлекли больше клиентов, нацеливаясь только на тех, у которых есть подобные потребности. Они делают нашу работу, намного легче. Есть масса особенностей в Campaign Monitor, которые бесполезны для всех, кроме сетевого проектировщика.

Фокусировка на основном рынке также помогает намного легче распространить информацию о вашем программном обеспечении. Теперь, когда у нас есть определенная аудитория, мы можем рекламироваться, в тех местах, где они часто бывают онлайн, издаем статьи, которые возможно, найдут интерес, и в общем строем сообщество вокруг нашего продукта.

— David Greiner, founder, Campaign Monitor

Расширяйтесь позже

У вас пока нет проблемы расширения

«Каким будет мое приложение, когда им будут пользоваться миллионы людей?»

Что? Ждите, пока это фактически не случится. Если вы достигли того, что огромное число людей, перегружают вашу систему — тогда ура! Это очень красивая проблема. Правда — подавляющее большинство сетевых приложений никогда не достигают этой стадии. И даже, когда вы достигните этого — обычно ничего страшного не происходит. У вас будет достаточно времени, чтобы приспособиться к проблеме. Плюс, вы будете иметь более реальные данные и эталонные тесты, чтобы выяснить к каким конкретно областям приложения требуются улучшения.

Например, у нас был запущен Basecamp на одном сервере в первый год. Поскольку мы шли с такой простой установкой, это потребовало всего неделю на осуществление планов. Мы не начинали с кластера в 15 боксов и не тратили месяцы на волнения о вычислениях

и на подсчеты.

У нас были проблемы? Нисколько. Но мы также осознали, что большинство проблем, которых мы боялись, действительно не существовало, с чем были согласны и наши клиенты. Будьте честны с ними, они поймут. Вспоминая прошлое, мы рады тому, что не стали делать «совершенную установку» с задержкой во многие месяцы.

В начале, сделайте основу продукта, вместо масштабируемости. Создайте большое приложение, а затем тревожьтесь о том, что делать, как только оно стало успешным. Иначе вы, возможно, расточаете энергию, время, и деньги, фокусируетесь на том, что может никогда не случиться.

Поверьте, это не большая проблема расширяться, когда в этом есть необходимость.

Вам придется снова все переделать, так или иначе

Дело в том, что всегда есть проблемы масштабируемости, никто не может сделать сразу с нуля то, что будут использовать миллионы потребителей. Снова придется переделывать почти каждый пункт проекта и архитектуры, чтобы обеспечивать масштабируемость.

— *Dare Obasanjo, Microsoft*

Делайте идейное программное обеспечение

Ваше приложение должно лавировать между потребностями

Некоторые люди считают, что программное обеспечение должно быть агностическим. Они говорят, что самоуверенно для разработчиков ограничивать особенности или пренебрегать запросами потребителей. Они говорят, что программное обеспечение должно всегда быть гибким, как только это возможно.

Мы думаем, что это ерунда. Лучшее программное обеспечение имеет свое виденье. Лучшее программное обеспечение лавирует между потребностями. Когда кто-нибудь использует программное обеспечение, они не только, ищут особенности, они ищут подход. Они ищут видение для решения своих задач. Решите, что такое — ваше виденье и идите с этим.

И помните, если им не нравится ваше виденье, есть масса других видений для других людей. Не преследуйте людей, так вы не сделаете их счастливыми.

Хороший пример — оригинальный wiki проект. Ward Cunningham и друзья сознательно раздели wiki на многие сущности, которые считались раньше целым документом. Вместо отнесения каждого изменения к определенному человеку или документу, они переместили многое в визуальное представления. Они сделали содержимое. Им было не важно, кто пишет содержимое или когда это было написано. И это сделало свое дело. Это решение поощряет общий смысл общества, и оно стало ключевым ингредиентом в успехе Wikipedia.

Наши приложения шли подобным путем. Они не пробуют охватить все для всех. У них есть свое отношение. Они ищут клиентов, которые являются фактически партнерами. Они обращаются к людям, которые разделили наше виденье. Вы или на автобусе, или от автобуса.

Выбор функций глава 5

Наполовину, но закончено

Создайте половину продукта, но законченный продукт

Остерегайтесь подхода в развитии сетевого приложения, в котором все готово, но вот что-то не работает и это что-то очень важное.

С Basecamp, мы начали только с секции сообщений. Мы знали, что это сердце приложения, так мы до поры до времени пренебрегли milestones, списками to-do, и другими элементами. Это позволило нам создать будущую основу при реальном использовании.

Начните быстро создавать приложение и позвольте ему приобретать инерцию. Затем вы можете начать добавлять функции и возможности твердой основе, которую вы построили.

Это не имеет значения

Только суть

Наш любимый ответ на вопрос «Почему вы не сделали это или почему вы не сделали то?». Всегда такой: «Поскольку это не имеет значения».

Когда мы запустили Campfire, мы слышали некоторые из этих вопросов от людей, впервые проверяющих продукт:

«Почему время показывается только каждые 5 минут? Почему нет времени в каждой линии чата?»

Ответ: Это не имеет значения. Как часто вам нужно отслеживать беседу с секундной или даже с минутной точностью? Конечно, не 95% времени. 5 минут вполне достаточно, поскольку какие-то меньшие промежутки времени — не имеют значения.

«Почему вы не позволяете форматирование текста жирным шрифтом, курсивным или выделение цветом в чатах?»

Ответ: Это не имеет значения. Если вам нужно сделать ударение на чем-нибудь используйте буквы верхнего регистра или поставьте несколько * (звездочек) вокруг слова или фразы. Эти решения не требуют дополнительного программного обеспечения, технической поддержки, дополнительной мощности обработки, и им не надо обучаться. Кроме того, форматирование в простом тексте чата не так важно.

«Почему вы не показываете общее число людей в комнате чата?»

Ответ: Это не имеет значения. Все имена внесены в список, так что вы знаете, кто есть в чате, но, какая разница, если это 12 или 16 человек? Если это не меняет ваше поведение, то это не имеет значения.

Хорошо если были бы эти функции? Безусловно. Но являются они сутью? Будут ли они востребованы? Нет. И вот почему мы их опустили. Лучшие проектировщики и лучшие программисты — не те, у кого лучшие навыки, или самые проворные пальцы, или не те, кто может сделать красивый макет в Photoshop, — это те, кто может определить, что имеет значение. Это те, кто понимает реальные выгоды от решения.

Большинство времени, которое вы тратите, расточается на том, что фактически бесполезно в работе. Если вы сможете сфокусировать работу и взгляд, на том, что имеет значение, вы достигнете производительности, которую никогда не воображали.

Начните ни с чем

Сделайте добавление новых функций, трудно осуществимой задачей

Секрет создания законченной половины продукта вместо недоделанной половины — снизить количество функций.

Каждый раз, когда вы решаете добавиьь новую особенность или возможность, вы принимаете ребенка. Вам приходится заставлять вашего малыша делать целую цепь событий (например, проект, выполнение, испытание, и т.п.). И каждый раз вы натыкаетесь на это.

Не будьте подпевалой

Сделайте добавление функций, трудно осуществимой задачей. Пусть каждая функция и особенность доказывает, что ее надо оставить в живых. Подобно «Бойцовскому клубу». Вы должны рассматривать только те функции и особенности, которые были протестированы в течение трех дней и за это время были востребованы максимально.

Если запрос на функцию постоянен, мы слушаем, но не действуем. Мы только знаем, что настало время взглянуть на это глубже. И только затем мы начинаем рассмотрение особенности в реальном окружении.

И что вы говорите людям, которые жалуются, когда вы не принимаете их идею функции? Напоминаем им, почему им нравится приложение в первоначальном виде: «Вам нравится это, потому что не делает 100 других вещей».

Скрытые затраты

Учитывайте скрытую цену новых функций и особенностей

Даже если функция уже работает, вам все еще ну36;но учитывает ее скрытые затраты.

Например, у нас есть запрос на то, чтобы добавить к Basecamp таблицу встреч. Это кажется достаточно простым, пока вы не рассмотрите это ближе. Подумайте обо всех различных элементах. Таблица встреч могла бы содержать: место, время, список людей, приглашения по электронной почте, календарь, интеграцию, документацию, поддержку, и т.п. Ради этого придется изменить соответствующие скриншоты, страницы тура, faq/помощь, условия обслуживания, интегрировать с другими возможностями и многое другое. Перед тем как добавить функцию, задумайтесь, сколько это с виду простое решение может принести головной боли.

Для каждой новой функции от вас потребуется:

- 1. Сказать «нет».
- 2. Вынудить функцию доказать свое значение.
- 3. Если снова «нет», уже конец. Если, «да», продолжайте…
- 4. Сделайте эскиз экрана/интерфейса.
- 5. Спроектируйте экран/интерфейс.
- 6. Закодируйте.
- 7-15. Испытание, испытание, испытание, испытание…
- 16. Проверка текста помощи, возможно, его нужно изменить.
- 17. Обновите ознакомительный тур продукта (если необходимо).
- 18. Обновите маркетинговую копию (если необходимо).
- 19. Обновите условия обслуживания (если необходимо).
- 20. Проверка, на то, какие предыдущие обещания были затронуты.
- 21. Проверка, на то, как это воздействует на общую структуру.
- 22. Запустите.
- 23. Затаите дыхание.

Можете ли управлять этим?

Создавайте то, чем можете управлять

Если вы запускаете партнерскую программу, должны ли вы управлять системой учета и выплат? Возможно, вы должны просто позволять людям зарабатывать без членских взносов, сообщений и отправке по почте проверок каждый месяц.

В состоянии ли вы отдать 1 гигабайт пространства бесплатно только потому, что Google это делает? Возможно, вы должны начать со 100 МБ, или только обеспечить место для платежных счетов.

Практичный совет: B1;оздавайте конструкции и услуги, которыми вы можете управлять. Легко давать обещания. Намного сложнее сдержать их. Сделайте то, что вы можете подтвердить фактически, организационно, стратегически, и материально.

Решение задач пользователей

Создавайте программное обеспечение для общих решений и поощряйте то, когда люди ищут собственные решения

Не навязывайте людям решения. Вместо этого пусть каждый будет генералом над программным обеспечением и сможет найти собственное решение проблемы. Предоставьте людям то, с помощью чего достаточно просто разрешить их собственные проблемы и найти собственный путь.

Когда мы создавали Ta-da List, мы намеренно пренебрегли многим. Нет никакой возможности отметить точно дату, нет никакой возможности, чтобы категоризировать элементы, и т.п.

Мы создавали инструмент, чистым и упорядоченным, позволяя людям творчески подходить

к решению задач. Люди, выясняли, как решить проблемы самостоятельно. Если они хотели добавить дату к элементу to-do, они могли добавить ее (точно так: April 7, 2006) непосредственно в сам элемент. Если они хотели добавить категорию, они могли добавить так [Книги], тоже непосредственно в сам элемент. Идеально? Бесконечно гибко? Да.

Если бы мы пробовали построить программное обеспечение, для того чтобы управлять этими сценариями, мы сделали бы менее полезный продукт для всех этих случаев.

Сделайте лучшую работу над основой проблемы. Люди найдут свои собственные решения и соглашения в пределах вашей общей структуры.

Забудьте о запросах функций

Пусть клиенты напоминают вам, что важно

Клиенты хотят, чтобы все было. Они будут присылать вам лавину запросов на новые функции. Просто проверьте форумы наших продуктов.

«Мы знаем, что это легко добавить» или «это можно сделать чуть лучше» или «это добавление займет всего несколько секунд» или «если вы добавите это, я заплачу в два раза больше» и так далее.

Конечно, мы не придираемся к этим людям. Мы поощряем их и слушаем. Любую свою продукцию мы начинаем делать с запроса клиента.

Но, мы упомянули, что ваш первый ответ на любой запрос должен быть — «нет». А что вы делаете со всеми этими запросами, которые к вам поступают? Где вы их храните? Как вы управляете ими? Вы этого не делаете. Просто читаете их, а затем отбрасываете.

Да, читайте их, отбросьте, и забудьте. Это звучит богохульно, но то, что важно будет продолжать поступать и дальше. То, что вам нужно помнить. То, что действительно является сутью. Не волнуйтесь об отслеживании и сохранении каждого запроса. Пусть ваши клиенты будут вашей памятью. Если это действительно стоящая функция, они будут напоминать вам, пока вы не сможете не забыть.

Как мы пришли к такому выводу? Когда мы запустили Basecamp мы отслеживали каждый запрос на функцию или особенности Basecamp, составляя to-do лист. Когда запрос был повторным, но от кого-то другого, мы обновляли список и ставили приоритет (II или III или IIII, и т.п.). Мы полагали, что будет один день, когда мы рассмотрим этот список и возьмемся за работу, за те запросы, у которых самый высокий приоритет.

Но, правда в том, что мы никогда не смотрели на эти списки снова. Мы уже знали, что нужно сделать, потому что наши клиенты постоянно напоминали нам об этом, повторяя одинаковые запросы снова и снова. Не было никакой потребности в списках или в уйме анализа, потому что это все происходило в реальном времени. Вы не можете забыть то, что важно, когда вы сталкиваетесь с этим каждый день.

И еще одна вещь: Просто, потому что несколько людей запрашивают что-нибудь, не означает, что вам придется это включать в разработку. Иногда нет ничего лучше того, чтобы промолчать лишний раз и поддерживать собственное виденье продукта.

Чего не хотят

Спросите людей, чего они не хотят

Большинство обзоров программных обеспечений и вопросов исследований сосредоточено вокруг того, что люди хотят от продукта. «Какая особенность отсутствует?» «Если вы могли добавить только одну вещь, чем это должно быть?» «Что сделало бы этот продукт, более полезным для вас?»

Что на обратной стороне монеты? Почему не спрашивают людей, чего они не хотят? «Если вы могли убрать одну особенность, что это было бы?» «Что вы не используете?» «Что делаете дольше всего?»

Больше «не» в вопросах. Иногда лучшее, что вы можете сделать для клиентов — опустить какую-нибудь функцию.

Новшества приходят из сказанного «нет»

Новшества приходят из сказанного «нет» тысячи вещам, чтобы убедиться в том, что мы не пойдем неправильным путем или не сделаем слишком много. Мы всегда смотрим на новые рынки, в которые могли бы войти, но, только говоря «нет», можем сосредоточиться на вещах, которые действительно важны.

— Steve Jobs, CEO, Apple (from The Seed of Apple's Innovation)

Процесс глава 6

Гонка в запуске программного обеспечения

Сделайте что-нибудь и идите быстро

Запуск программного обеспечения — лучший способ добиться инерции, поучаствовать в ралли командой, и отсеять идеи, которые не работают. Это должно быть приоритетом номер один в первый же день работы.

Хорошо сделать меньше, опустить детали, и сократить процесс так, чтобы выпустить программное обеспечение быстрее.

Как только вы выпустили программу, вы будете вознаграждены тем, что будете знать более точную перспективу как дальше продолжать развитие. Описания, каркас, даже html макеты, только приближают вас к этому. Запустите программное обеспечение в реальную работу.

С запуском программы в реальную работу, вы добирается до истинного понимания того, как она должна работать. Вы избегаете бурных разговоров, эскизов и длинных текстов, которые отдаляют вас от сути. Вы осознаете, какие мысли были тривиальны, а какие критически неприемлемы

Повторение и свобода

Работайте итерационно

Не ожидайте того, что будете все понимать и делать правильно с первого раза. Пусть приложение растет и общается с вами. Позвольте ему трансформироваться и эволюционировать.

Нет никакой необходимости отправлять веб-программы в плавание совершенными. Проектируйте интерфейсы, используйте их, анализируйте, а затем начинайте снова.

В отличие от банковских дел по получению аванса, итеративный процесс позволяет вам продолжать принимать информированные решения, так как вы идете в ногу с разработкой. Плюс, вы получаете активно работающее приложение. Результат — реальная обратная связь и реальное понимание того, что требует вашего внимания.

Повторения приводят к свободе

Если вы знаете, что собираетесь переделать все снова, вам не нужно нацеливаться на совершенствования при первой попытке. Это знание — большой фактор мотивации, и способ проверить свои идеи фактически и при необходимости откорректировать их.

От идеи к реализации

Перейдите от мозговых штурмов — к эскизам — к HTML — к кодированию

Вот процесс Get Real, который мы используем:

Мозговой штурм

Начинайте с идеи. Что этот продукт собирается делать? Для Basecamp, мы смотрели на свои собственные потребности. Мы хотели сделать проектные модификации. Мы хотели, чтобы клиенты участвовали в проекте. Мы знали, что проекты должны иметь milestones. Мы хотели централизовать архивы, так чтобы люди могли легко рассматривать старый материал. Мы хотели сделать большие картинки в проектах, видимые с большого расстояния. Вместе, те предположения, и несколько другие, служили нам основой.

Эта стадия состоит из вопросов. Что приложению нужно делать? Как мы будем знать когда это полезно? Что точно мы собираемся сделать? Это высокоуровневые идеи, а не обсуждение деталей, таких как расстояния в пикселях от кнопки до чего-то еще. На этой стадии видны только значимые детали.

Бумажные эскизы

Быстрые, простые эскизы — это самый экономичный способ начать проектирование. Выводите свои идеи на бумагу, пусть и небрежным почерком. Цель этой стадии превратить мысли и понятия в набросок интерфейса проекта. Этот шаг — экспериментирование, в котором нет ошибок или неправильных ответов.

HTML макеты

Делайте html версии того, что изображено на эскизах. Получите то, что уже будет отдаленно напоминать программу на экране.

Для Basecamp мы сначала сделали макет написания сообщения и макет редактирования сообщения. И дальше отталкивались от этих макетов.

Не пишите никакого программного кода. Просто стройте макеты с использованием html и css.

Закодируйте это

Когда макет демонстрирует достаточное количество необходимой функциональности, можно приступать к программированию.

В процессе этого, помните, что вас ждут многократные повторения и оставляйте проект гибким. Не стесняйтесь отбрасывать специфические шаги и начинать их потом. Главное сразу исключить плохой и не продуманный код.

Избегайте настроек

Примите решение о деталях

Вы сталкиваетесь с ограничением: сколько сообщений должно быть на странице? Ваша первая мысль сделать выбор 25, 50 или 100. Это легкий выход. Просто примите решение, как сделать лучше. И выберите одно число.

Настройки — уход от пути принятия жестких решений

Чтобы выбрать в программе лучшие решения — для этого есть вы. Не перекладывайте принятие решений на плечи клиентов. Для клиентов экраны настроек с бесконечным количеством выбора — головная боль. Клиенты не должны думать о каждой мелочи, за это ответственны вы.

Настройки — зло, потому что они раздувают программное обеспечение и требуют больше кода. А в реальности очень часто настройками никто даже не пользуется. Настройки подразумевают, что вы мало знаете о том, как должны быть расположены блоки на странице, сколько сообщений должно быть выведено на страницу и т.п.

Сделайте выбор

Примите простые решения. Это — то, что мы сделали в Basecamp. Число сообщений на страницу составляет 25. На странице краткого обС7;ора показаны последние 25 элементов. Сообщения сортируются в хронологическом порядке. Пять, последних проектов показываются в dashboard. Нет вариантов выбора.

Да, возможно, сделали плохой выбор. Но если это так, то люди будут жаловаться и всегда можно будет выбор подкорректировать. Getting Real — это возможность измениться на лету.

«Сделано!»

Решения временны

Сделано! Это волшебное слово. Когда вы уже сделали — вы получили опыт и можете идти дальше.

Но подождите, а что если вы сделали что-то неправильно? Это плохо. Но это не нейрохирургия, это — сетевое приложение. Нет ничего страшного. Не нужно только замедлять процесс анализом ошибок. Вместо этого оцените важность и идите дальше.

Признайте, что это решение было временным. Признайте, что ошибки будут и дальше. Реализуйте только возможность быстро исправлять свои ошибки.

Тестируйте в диких условиях

Испытывайте ваше приложение в реальных условиях

Нет никакой замены реальных людей, использующих ваше приложение в действительности. Получите реальные данные. Получите реальную обратную связь. Затем улучшайте, основываясь на этой информации.

Тесты в лабораториях не отражают действительность. Если вы стоите над кем-то и проверяете, вы не получите точного отражения работы. Когда действия человека записываются на камеру, он все равно осторожен. Когда кто-нибудь другой наблюдает, люди особенно осторожны в том, чтобы не наделать ошибок.

Вместо этого, тестируйте в реальной работе, в реальном технологическом процессе, где реальные люди подвергнут функционал нагрузкам и заполнят приложение реальными данными. Только так вы получите реальные результаты.

Не нужно никаких бета-версий. Версия должна быть одной. Бета-версия допускает поверхностное использование, а реальная версия получает полное и реальное использование.

Сокращайте время

Разделяйте

Сокращайте время разработки, разбивая месяцы на недели. Если проект требует 12 недель работы, разбейте его на 12 небольших кускоС2;. Если проект требует 30 часов работы, разбейте Cd;а куски по 6-10 часов и делайте каждый шаг за один раз, за один день.

Вы сталкиваетесь с большой проблемой, которую уже не в состоянии держать в голове. Разделите ее на кусочки, каждый из которых сможете обдумать с легкостью и решить за один шаг.

Организация глава 7

Единство

Не разделяйте лишний раз

У многих компаний такие направления как проектирование, развитие, копирайтинг, поддержка, маркетинг разделены на отделы. Такая специализация по отделам, конечно, имеет свои преимущества, но также создает ситуацию, когда сотрудники видят только свой маленький мир, а не полный контекст веб-приложения.

Как можно больше смешивайте команду по различным специализациям, так чтобы наладить диалог между всеми в процессе работы. Не позволяйте чему-то теряться из-за отсутствия диалога в команде. Пусть копирайтеры работают над текстом вместе с проектировщиками. Убедитесь, что разработчики видят, как работает поддержка.

Еще лучше нанять людей с разными талантами, которые они будут применять в течение всей работы над проектом. Результатом будет более гармоничный продукт.

Единое время

Людям нужно время, чтобы постоянно добиваться своего

Сотрудники 37signals находятся в четырех городах и в восьми часовых поясах. От Прово, штат Юта до Копенгагена, Дания. Есть только 4-5 часов в течение дня, когда мы работаем вместе. Когда наша команда спит, Давид, который находится в Дании, работает.
Мы работаем, Давид спит. Это дает нам только половину дня для совместной работы.

Можно предположить, что и время работы, и эффективность от работы сокращается вдвое. Но это нет так.

Известно, что многие люди предпочитают работать рано утром или поздно вечером — это время когда их не беспокоят, это время самое продуктивное для них. Это время становится таким, потому что никто не прерывает работу, и никто не отвлекает.

Поэтому можно установить правило: сделать половину рабочего дня единым временем для работы. В это время не отвлекаться на телефонные звонки, чтение почты, ненужное общение с коллегами и т.д.

Избегайте любых перерывов во время единого времени работы. Каждый перерыв снижает производительность и отвлекает настолько, что потом опять необходимо вникать в работу. Это все снижает продуктивность и увеличивает количество часов, требуемых для выполнения работы.

Встречи ядовиты

Никаких встреч

Вам действительно нужны встречи? Встречи обычно возникают, когда что-то не достаточно ясно. Вместо встречи, попытайтесь упростить обсуждение и воспользуйтесь мессенджером или Campfire. Минута встречи крадет минуту реальной работы. Поставьте себе цель — избегать встреч.

Нет страшнее яда для производительности, чем встреча. У этого несколько причин:

1. Встречи разбивают рабочий день на отдельные куски и этим прерывают ваш естественный технологический процесс.
2. Встречи — это всего лишь слова и абстрактные понятия.
3. Встречи несут безнадежно мало полезной информации в минуту.
4. На встречах часто присутствует один слабоумный, который неизбежно заставляет всех тратить на него общее время.
5. У встреч есть повестки дня, такие неопределенные, что часто никто действительно не уверен в том, что надо обсуждать.
6. Встречи требуют большой подготовки, чего почти никто не делает.

Если все-таки вы не можете избежать встречи (что должно быть очень редким событием), воспользуйтесь простыми правилами:

1. Встреча должна быть не более 30 минут. Установите таймер, после 30 минут прерывайте встречу.
2. Пригласите как можно меньше участников.
3. Никогда не участвуйте во встрече без ясной повестки дня.

Находите и празднуйте маленькие победы

Выпустите что-нибудь сегодня

Самое главное в разработке программного обеспечения — мотивация. Сначала мотивация, потом возможности. Только с мотивацией можно сделать что-то действительно хорошо.

Долгие, затянутые циклы разработки — убийцы мотивации. Это слишком долгое время между празднованием побед. Быстрые победы — факторы мотивации. Если вы допускаете длинные циклы разработки — вы убиваете мотивацию. И это может убить ваш продукт.

Среди многих месяцев разработки посвятите один день в неделю для маленькой победы. Спросите себя: «Что можно сделать и выпустить за 4 часа?». Затем сделайте это.

Это может быть:

1. Новая простая особенность
2. Маленькая доработка существующей особенности
3. Написание некоторого текста помощи, чтобы сократить бремя поддержки
4. Удаление нескольких полей форм, которые вам действительно не нужны

Когда вы делаете что-то за 4 часа до конца, вы можете каждый раз праздновать победу. Это создает хорошее моральное состояние, увеличивает мотивацию и подтверждает, что команда движется в правильном направлении.

Персонал глава 8

Нанимайте меньше. Нанимайте позже

Добавляйте медленее, чтобы двигаться быстрее

Нет необходимости становиться большими рано — так же как и поздно. Даже если у вас есть доступ к сотне лучших людей, это все равно плохая идея попробовать нанять их всех одновременно. Нет никакого способа, с помощью которого вы смогли бы сразу объединить столь много людей в гармоничный коллектив. Вы неизбежно столкнетесь с головной болью обучения, межличностными конфликтами, непониманием в общении, людьми с разными направлениями мышления и многим другим.

Так что не нанимайте. Серьезно, не нанимайте никого. Взгляните на это с другой стороны — действительно ли работа, которая вас отягощает, так необходима? Что случиться, если вы просто её не сделаете? Можете ли вы решить проблему с данной частью системы другим способом?

Каждый раз когда Jack Welch (former CEO of GE) увольнял кого-либо, он не нанимал человека взамен сразу. Он хотел посмотреть как долго GE может обходиться без данной персоны и его роли. Конечно, мы не призываем увольнять людей, чтобы проверить эту теорию, но мы думаем, что Jack настаивал на чём-то вроде: «Вам не нужно столько людей, сколько вам кажется необходимым».

Если нет никакого другого варианта — рассмотрите возможность найма нового человека. Но вы должны точно знать кто вам нужен, как вы введёте его в работу и как много головной боли обретёте.

Закон Брукса

Привлечение людей к просроченному проекту задержит его ещё больше.

—Fred Brooks

Программирование и «Реквием» Моцарта

Хороший одиночный программист, работающий над отдельной задачей, не имеет над собой проблем координации работы и общения. Пять программистов, работающих над той же задачей, обязаны общаться и координировать свои действия между собой. Это занимает много времени... Основная проблема с использованием группы посредственных программистов вместо парочки хороших состоит в том, что не имеет значение как долго они работают, они никогда не создадут что-то так же хорошо, как это делают великие программисты. Пять Антонио Сальери не смогут создать «Реквием» Моцарта. Никогда. Даже, если они будут работать над этим 100 лет.

—Joel Spolsky, software developer, Fog Creek Software (from Hitting the High Notes)

Проверяйте

Назначайте потенциальным работникам испытательный срок

Одно дело рассматривать портфолио, резюме, примеры кода или предыдущую работу кого-либо. Другое - поработать с ним в реальных условиях. Всегда, когда это возможно, назначайте потенциальному новому члену команды тестовое задание.

Прежде чем мы нанимаем кого-либо, мы даем ему на пробу небольшой проект. Мы смотрим, как он ведёт проект, как общается, как работает и т.д. Наблюдение за тем, как человек проектирует или кодирует несколько страниц, даст вам множество информации для понимания того, подходит он вам или нет.

Сроки для подобных вещей могут быть сжатыми, но даже если это задание на 20 или 40 часов - это лучше, чем ничего. Соответствует ли вам человек или нет - это будет очевидно. И если нет - то обе стороны оберегут друг друга от множества проблем и рисков, проиграв возможную ситуацию наперёд.

Начните с малого

Попробуйте для начала дать небольшое тестовое задание. Не надо набрасываться сразу со всей вашей работой. Дайте вашему новому виртуальному помощнику (virtual assistant, VA) один или два проекта для проверки и посмотрите, как налаживается с ним взаимопонимание. В самом начале может показаться легким закрыть глаза на потенциальные проблемы, но помните - это проверочный забег.

—Suzanne Falter-Barns, author/creativity expert
(from How To Find And Keep The Perfect VA)

Не по словам, а по делам

Ищите потенциальных работников среди разработчиков проектов с открытым кодом

Типичный метод приёма на работу технических специалистов — тот, при котором рассматривается научная степень, резюме и т. д. — слаб по многим причинам. Так уж важно кто где учился или средний бал успеваемости? Вы реально поверите тому, что написано в резюме или рекомендациях?

Проекты с открытым кодом — находка для того, кто ищет техперсонал. Они дают вам возможность следить длительное время за работой того или иного специалиста.

Это значит, вы можете оценивать людей по-сделанному, а не по-сказанному. Вы сможете принимать решение основываясь на действительно важных факторах:

- **качество работы**
  Некоторые программисты могут красочно «заливать» о своих способностях, но при этом «съезжать с темы» когда доходит до дела. Участие человека в открытых проектах покажет суть его умений и опыта.
- **воззрения**

Программирование это постоянное принятие решений. Огромного их количества. Решения зависят от воззрений, ценностей и идеалов. Присмотритесь к тем решениям, которые принял потенциальный кандидат при разработке кода, тестировании, в спорах, чтобы понять, в какой мере его представления соответствуют вашим. Если эта мера мала, вам будет трудно принимать решения вместе.

- **уровень увлеченности**
  Определенно, участие в открытых проектах требует хоть малость увлеченности. Иначе, зачем человек тратит время сидя за монитором? Степень занятости в открытых проектах часто показывает на сколько человек действительно любит программировать.
- **умение довести дело до конца**
  Ни самые мудрые и правильные воззрения, ни огромнейшая увлеченность не стоят ничего, если кандидат не может довести дело до конца. К сожалению, это дано не всем. Ищите тех кто может. Нанимайте человека, который доводит дело до конца, во что бы не стало; но в тоже время, способного находить прагматичные компромиссы.
- **умение работать в коллективе**
  Работая с кем либо долгое время, проходя через взлёты и падения, стрессы и «расслабоны» вы увидете реальный характер человека. Даже не обращайте внимания на тех, кто не обладает достаточной сдержанностью, или щедро наделён несносными манерами.

Когда речь идет о программистах, мы нанимаем только тех, кого мы знаем по проектам с открытыми исходниками. Считаем остальные подходы несостоятельными. Мы пригласили Джеймиса заметив его участие в разработках сообщества Ruby. Он выделялся среди остальных. Не было необходимости рассматривать второстепенные факторы, поскольку мы могли оценить его работу по главному критерию — качеству.

Не переживайте если какие-либо внештатные увлечения отвлекают ваших сотрудников от их основной работы. Перефразируя избитое выражение: если хочешь, чтобы дело было сделано хорошо, доверь его самому занятому человеку. Дэвид и Джеймис самые активные разработчики Rails и все еще успешно справляются с техническими вопросами в 37signals. Вашей команде нужны те, кто обожает программирование и умеет довести начатое до конца.

Увлеченные программированием

Что вы больше всего хотите от недавно нанятого человека? Конечно увлеченности своим делом. И нет лучшего способа увидеть это, чем посмотреть на участие человека в открытых проектах.

—Jarkko Laine, разработчик ПО
(из «Уменьшите риски, нанимайте опенсорсеров»)

Нанимайте эрудитов

Быстро обучающиеся универсалы лучше закостенелых авторитетов

Мы никогда не наймем кого-либо, чья профессия называется «разработчик структуры системы программного обеспечения». Это уж слишком. В маленьких командах, таких как наша, нет смысла в подобного рода узких специалистах.

В маленьких командах нужны мастера «на все руки». Дизайнеры, умеющие вести

техническую документацию. Программисты, сведущие в вопросах дизайна. Каждый должен себе представлять какой должна быть структура системы (какой бы смысл вы бы не вкладывали в это понятие). Каждому нужно мыслить системно. Каждый должен уметь общаться с клиентами. И, в случае необходимости, каждый должен уметь, резко «дав по тормозам», развернуться контролируемым заносом. Запомните, маленьким командам иногда надо изменить направление и сделать это быстро. Вам нужен тот, кто сможет учиться, привыкать, приспосабливаться, а не закостенелый брюзга, специализирующийся на чём-то одном.

Неподдельный энтузиазм

Берите счастливых середнячков, а не разочаровывающих гуру

Энтузиазм. Свойство, не передающееся даже самым умелым притворством. Пришло время нанять кого-то в команду? Не думайте, что нам нужен гуру или хорошо известный (в тесных кругах) специалист. Такие всё равно чаще всего выступают в роли «свадебных генералов». Счастливый специалист средней руки лучше раздражённого эксперта.

Берите того, у кого есть энтузиазм. Того, кому вы доверите остаться самому и доделать работу. Того, кто страдал в больших и неповоротливых компаниях и жаждет новой обстановки. Того, кому нравиться делать то же что делаете вы. Того, кто ненавидит то же, что и вы. Того, у кого учащается дыхание, когда он взбирается на борт вашего корабля.

Дополнительные баллы тем, кто спрашивает

Обратите внимание, задает ли соискатель вопросы о вашем проекте. Увлеченные программисты хотят максимально разобраться в проблеме и готовы сразу предложить потенциальные решения или улучшения, — и поэтому много спрашивают. Разбирая вопросы, вы также поймете, что проект можно реализовать сотнями способов и очень важно точно, с предельной чёткостью выразить своё видение того, как будет работать ваше веб-приложение. По мере того, как вы углубляетесь в детали, вы всё больше чувствуете на сколько опыт и характер человека соответствуют вашей команде.

—Eric Stephens, BuildV1.com

Мастера Слова

Нанимайте хороших писателей

Если вы задумываетесь над тем, какого рода специалиста можно еще пригласить на не занятое место, — наймите того, кто лучше других умеет вести документацию. Не важно кто он — дизайнер, программист, специалист по продажам или кто-то еще, его умение хорошо писать окупится с лихвой. Понятная, последовательная документация ведет к понятному и последовательному коду, дизайну, письмам, объявлениям и т. д.

Хороший писатель не просто тот, кто грамотен. Хорошие писатели знают как донести суть.

Они делают сложное — понятным. Они умеют посмотреть на вещи «непонимающим» взглядом. Они знают что лишнее. Их мысли ясны. И они те, кто вам нужен.

Светлый Разум

Умение хорошо писать показатель того, на сколько организован человек; способен ли он упорядочивать, систематизировать и подавать информацию так, чтобы другим было легче её понять. Это сказывается на коде, общении, чатах (для коллективов которые работают удаленно) и даже на таких сокровенных понятиях как профессионализм и надежность.

—Dustin J. Mitchell, разработчик (из Signal vs. Noise)

Письмо упорядочивает мысли

Отчетливое письмо делает отчётливее и мысли. Вы не знаете что именно вы знаете пока вы не выразите это. Умение понятно писать — это, в некоторой степени, черта характера. Вместо того, что упрощать что-то для себя, упростите это для своих читателей.

—Michael A. Covington, профессор теории вычислительных систем Университета Джорджии
(из Как правильнее писать, четче мыслить и изучать сложное просто)

Создание интерфейса chapter 9

Сначала - интерфейс

Создавайте дизайн интерфейса до того как начнете программировать

Слишком много приложений создаются с подходом "сначала программируем". Это неудачная идея. Программирование - самое сложное в создании приложения, а это значит, что и самая дорогая. И создав код, вам будет сложно его изменить. Вместо этого начинайте с дизайна интерфейса.

Дизайн относительно легко изменять. Бумажный набросок дешев и его легко изменить. html-наброски тоже довольно просто изменить или просто выбросить. Но в отношении программирования это неверно. Подход "сначала дизайн" позволит вам быть гибкими. Подход "сначала программирование" ограничивает вас и приводит к дополнительным затратам.

Другая причина для того, чтобы начинать с дизайна в том, что интерфейс и есть продукт. Вы продаете людям то, что они видят. И если вы оставите интерфейс "на потом", огрехи будут заметны.

Мы начинаем с интерфейса, и можем "пощупать" приложение с самого начала. И интерфейс постоянно пересматривается в процессе разработки. Понятен ли он? Прост ли в использовании? Позволяет ли легко решить проблему? Верно ответить на эти вопросы вы можете только если уже имеете реальный интерфейс. Подход "сначала дизайн" позволяет вам оставаться гибкими и подталкивает к ответам на эти вопросы как можно раньше, вместо того что бы оставлять их "на потом".

Оранжевая ручка, с которой начался Blinksale

Как только я понял, что готовые приложения для выставления счетов меня не устраивают, я решил нарисовать на бумаге, как я представляю такое приложение. Я взял оранжевую ручку, потому что больше в тот вечер было нечем рисовать, и через несколько часов три четверти будущего приложения были готовы. Я показал всё своей жене, Рейчл, которая как раз гладила и спросил её, что она думает по этому поводу. И она ответила с улыбкой: "Тебе надо сделать это. Правда."

Следующие две недели я дорабатывал дизайн, и сделал статические наброски для всех страниц первой версии того, что потом стало называться Blinksale. Мы никогда не делали никаких каркасов кроме тех набросков оранжевой ручкой, и то, что мы перешли сразу к html-страницам, подстёгивало нас, так как проект становился более реальным, хотя в то время мы и не знали что именно происходит.

Как только html-макеты были готовы, мы рассказали идею нашему разработчику, Скотту. То, что большая часть дизайна была уже создана, было весьма кстати. Во-первых, это дало Скотту реальной представление о направлении, в котором мы движемся, и вовлекло его. Это было больше чем идея, это была реальность. Во-вторых, это помогло нам подсчитать, сколько усилий и времени потребуется от Скотта, чтобы превратить дизайн в работающее приложение. Когда вы финансируете проект с самого начала, то чем раньше вы сможете предсказать бюджет тем лучше. Дизайн пользовательского интерфейса стал нашим мерилом для границ проекта. И последнее - дизайн интерфейса служил нам для того, чтобы напомнить нам о том, для чего предназначено приложение, по мере того, как продвигалась разработка. Каждый раз как появлялся соблазн добавить новые возможности, мы не могли просто сказать "А давайте-ка добавим вот это и ещё это!". Мы должны были бы вернуться к дизайну и спросить себя, где надо добавить новую возможность, и если для неё не было места, мы её не добавляли

—Josh Williams, основатель, Blinksale

Дизайн от эпицентра

Начинайте с самого важного на странице и двигайтесь вширь

Дизайн от эпицентра фокусирует внимание на том, что наиболее важно на странице, а затем обращается к менее важному. Это значит, что сначала вы игнорируете то что находится кругом - навигацию, закладки, "подвал" страницы, цвета, логотип и так далее. Вместо этого, вы начинаете с эпицентра и сначала разрабатываете наиболее важную часть страницы.

Без чего страница аобсолютно не может быть - это без эпицентра. К примеру, если вы разрабатываете страницу, которая показывает пост в блоге, пост - это и есть эпицентр. Не категории в сайдбаре, не заголовок страницы, не форма комментариев внизу, а именно пост. Без поста эта страница бессмысленна.

Только когда эпицентр готов, можно подумать о втором по критичности элементе на странице. После второго вы можете перейти к третьему и так далее. Это и есть дизайн "от эпицентра".

Дизайн от эпицентра избегает традиционной модели"давайте построим раскладку, потом засунем туда контент". В этом процессе строится общая форма страницы, потом добавляется навигация, потом маркетинговое барахло, и потом, наконец, основная функциональность, то, для чего страница предназначена, впихивается на оставшееся место. Это перевернутый

процесс в котором наиболее приоритетные вещи откладываются на потом.

Дизайн от эпицентра переворачивает этот процесс и позволяет вам сфокусироваться на том, что важно, в первый день. Основное - сначала, дополнения - потом. В результате получается более дружественный, ориентированный на пользователя, удобный в использовании экран для клиентов. Плюс, это позволяет вам начать диалог между дизайнером и разработчиком прямо сейчас вместо того чтобы ждать пока все аспекты страницы будут проработаны.

Три состояния программы

Делайте дизайн для обычного, пустого, и ошибочного состояния страницы.

Для каждого экрана вы должны рассмотреть три состояния:

- **Обычное**
  Экран, который люди видят каждый день при работе с приложением, заполненным данными.
- **Пустое**
  Экран, который люди видят при первом запуске, и ещё не успели ввести данные.
- **Ошибка!**
  Экран, который люди видят, когда что-то идёт не так.

Обычное состояние - это элементарно :) Это экран, где вы проводите бОльшую часть вашего времени. Но не забывайте инвестировать время и в другие состояния (читайте следущие эссе, чтобы узнать больше об этом).

С самого начала

Создавайте ожидания, продумав то, какой опыт получит пользователь при первом запуске

Игнорировать состояние "чистого листа" - одна из самых больших ошибок, которую вы можете сделать. "Чистый лист" - это первое впечатление от вашего приложения, и у вас не будет второго шанса его создать... ну, вы знаете.

Проблема в том, что когда вы разрабатываете интерфейс, он обычно заполнен данными. Дизайнеры всегда заполняют шаблоны данными. Каждый список, каждый пост блога, каждое поле, все тёмные углы и щелки в шаблонах заполнены данными. И это значит, что экран выглядит и работает великолепно.

Однако обычное состояние приложения - когда оно не не набито данными. Когда кто-то регистрируется, он начинает с чистого листа. Более похоже на блог, который надо заполнить - общий вид неясен, пока люди не введут свои данные: посты, ссылки, комментарии, часы, информацию на сайдбаре, или ещё что-то.

К сожалению, пользователь решает, достойно ли приложение внимания, на этапе "чистого листа" - — на этапе, когда количество информации, дизайн и содержимое по которым судят

полезность приложения в целом, минимальны. Если вам не удаётся создать адекватное состояние "чистого листа", люди не знают, чего им не хватает, потому что им не хватает всего.

Пока бОльшая часть дизайнеров и разработчиков всё ещё принимает это состояние за само собой разумеющееся. Они не могут разработать состояние "чистого листа", потому что когда они разрабатывают или пользуются приложением, оно полно данных, которые они ввели для тестирования.Они даже не встречаются с состоянием "чистого листа". Конечно, они могут войти в качестве нового пользователя пару раз, но бОльшая часть их времени проходит в плавании по приложению, заполненному данными.

Что вы должны включать в действительно полезный "чистый лист"?

- Используйте его как возможность вставить краткий обучающий курс и подсказки по использованию приложения.
- Дайте снимок экрана, заполненного данными, чтобы люди знали чего ожидать (и с чем им придётся иметь дело).
- Объясните, с чего начать, на что будет похож экран и так далее.
- Отвечайте на основные вопросы тех, кто пришёл первый раз: Что это за страница? Что я сейчас делаю? Как будет выглядеть заполненный экран?
- Создавайте ожидания, помогите избежать разочарования, страхов и смущения.

Первые впечатления жизненно важны. Если вам не удаётся создать продуманное состояние "чистого листа", вы создадите негативное (и неверное) впечатление от вашего приложения или сервиса.

У вас никогда не будет второго шанса...

Ещё один аспект интерфейса Mac OS X, на который, я думаю, сильно повлияло мнение Стива Джобса (Steve Jobs) - это установка и первый запуск системы. Я думаю, Джобс в курсе важности первого впечатления. Я думаю, Джобс смотрит на то как выглядит первый запуск системы и обдумывает его. Это, возможно, одна тысячная общего опыта работы с машиной, но это самая важная тысячная, потому что это первая тысячна и она создаёт ожидания и начальное впечатление.

—*John Gruber, автор и разработчик (из интервью с John Gruber)*

Защищайтесь

Продумайте дизайн для нештатных ситуаций

Посмотрим правде в глаза: неполадки обязательно будут происходить. Как бы тщательно вы не проектировали приложение, сколько бы времени не посвятили тестированию, у клиентов все равно будут случаться проблемы. Так как вы будете обрабатывать эти поломки? С помощью оборонительного дизайна.

Оборонительный дизайн похож на контраварийное вождение. Так же, как водителям всегда нужно быть начеку – бывают скользкие дороги, безрассудные водители и другие опасные сценарии, – создатели сайтов должны постоянно следить за источниками проблем, которые могут приводить к растерянности и расстройству посетителей . Хорошая защищенность сайта определяет, насколько успешным для клиентов будет опыт использования сайта.

Мы могли бы многое рассказать об оборонительном дизайне – хватило бы на отдельную книгу. Собственно, мы ее уже написали. Книга называется “Оборонительный дизайн для

сети”(Defensive Design for the Web), и послужит хорошей подмогой любому, кто хочет улучшить сообщения об ошибках и другие критические точки.

Помните: ваше приложение может работать отлично в 90% случаев. Но если вы бросаете пользователей на произвол судьбы там, где ваша помощь им нужна – они вряд ли это забудут.

Содержание важнее целостности

То, что имеет смысл «здесь», может потерять его «там»

Должны ли действия задаваться кнопками или ссылками? Это зависит от действия. Должны ли даты выбираться из списка или из сетки календаря? Это зависит от того, где будут показаны даты и какой временной период они задают. Нужны ли все линки навигации на каждой странице? Нужна ли всюду строка поиска? Нужно ли повторять колонтитул на каждой странице? Правильный ответ: “смотря по обстоятельствам.”

Вот почему содержание гораздо важнее целостности. Это нормально — сделать дизайн непоследовательным, если в этом есть смысл. Давайте людям то, что имеет смысл. Давайте то, что им нужно, и избавте от того, что лишнее. Уместность лучше последовательности.

Разумное непостоянство

Целостность не обязательна. Годами студентам твердят, что целостность интерфейса — одно из важнейших правил дизайна интерфейсов. Возможно, это справедливо для традиционного ПО, но не для веб, где каждый сайт отличается от другого и посетители могу легко и быстро перемещаться по ним.

В Creative Good, мы называем это “разумным непостоянством”: на каждой странице пользователи видят именно то, что им нужно на том шаге, на котором они находятся. Добавлять избыточные элементы навигации лишь для того, чтобы сохранить целостность сайта, просто глупо.

—Mark Hurst, основатель Creative Good и создатель Goovite.com
(из The Page Paradigm)

Дизайн интерфейсов — это копирайтинг

Каждая буква имеет значение

Дизайн интерфейсов — это копирайтинг. Лучшие интерфейсы написаны. Если вы думаете что каждый пикD1;ел, каждая иконка, каждый шрифт имеет значение, вы поверите и в значимость каждой буквы. Когда вы описываете свой интерфейс, всегда смотрите на него глазами пользователей. Что они должны знать? Как это объяснить лаконично и доходчиво?

Назовёте ли вы кнопку «ОК» или «Сохранить» или «Обновить»? «Добавить» или «Создать»? Это копирайтинг. Напишите ли вы три предложения или пять? Объясните ли вы на общих примерах или со всеми подробностями? Пометите ли вы записи как «Новые», «Недавно

обновлённые» или «Измененные». «Всего новых сообщений: 5» или «5 новых сообщений»? «5» или «пять»? «сообщений» или «постов»? Всё имеет значение.

Вы должны говорить тем же языком, что и ваша аудитория. Тот факт, что вы пишете приложение для веб, еще не значит что будет уместен технический жаргон. Думайте о своих клиентах и о том, что значат для них эти кнопки и слова. Не используйте малознакомых сокращений или слов. Не выражайтесь словами директора на совещании. Будьте кратки и вежливы. Скажите главное, и не более.

Хорошая документация — это хороший дизайн. Это редкое исключение когда слова не соответствуют дизайну. Иконки, названия, формы примеров, надписи на кнопках, пошаговые инструкции, понятное объяснение правил возврата денег — все это дизайн интерфейсов.

Единый интерфейс

Объединяйте настройки с основным интерфейсом

Страницы, содержащие системные настройки, списки пользователей и т.п., зачастую крайне неудобны и ужасно выглядят. Всё потому, что большая часть времени тратится на разработку основного интерфейса.

Не создавайте отдельного интерфейса для настроек, просто встройте его функции в основной.

Поддерживая сразу два интерфейса вы делаете лишнюю работу. Всё равно, что платить один и тот же налог дважды. Чем чаще вы вынуждены повторяться, тем выше шансы ошибиться. Напротив, чем меньше страниц вам приходиться поддерживать, тем легче и проще для вас и тем лучше для вашего клиента.

Нет разделению интерфейсов

Приложение - единое целое. Всё что может быть изменено, добавлено или настроено, должно изменяться, добавляться и настраиваться напрямую из приложения. Когда мы видим именно то, что видят наши клиенты - нам гораздо проще разобраться в любых возникших проблемах. Нашему клиенту не нужно пользоваться несколькими разными интерфейсами. В данную секунду он работает с деловым расписанием, в следущую он захочет создать учётную запись для нового сотрудника. Зачем ему переключаться в другое приложение? Пользуясь одним понятным и последовательным интерфейсом он освоится гораздо быстрее.

—Эдвард Ниттел, Директор по продажам и маркетингу, KennelSource

---

Код глава 10

Меньший объем программы

Сохраняйте код как можно более простым

Вам кажется, что имея в два раза больше кода, ваша программа будет только вдвое сложнее.

На самом деле, **каждый раз, когда вы увеличиваете объем кода, сложность программы возрастает экспоненциально**. Каждое маленькое дополнение, каждое изменение, каждая взаимозависимость и каждое предпочтение имеют каскадный эффект. Продолжайте безбоязненно добавлять код, и вы получите страшный Большой Ком Грязи – до того, как это заметите.

Как бороться с этой сложностью – уменьшением объема программы. Уменьшение объема программы значит меньше функций, меньше кода, меньше отходов.

Главное здесь – переформулировать любую сложную задачу, требующую много кода, в простую задачу, которая требует кода намного меньше. Возможно, вы уже не будете решать в точности ту же задачу – это нормально. Решить 80% первоначальной задачи, затратив 20% усилий – это большой выигрыш. Первоначальный вариант задачи обычно никогда не является настолько важным, чтобы затрачивать в пять раз больше времени на ее решение.

Меньший объем программы значит, что вам не придется теряться в догадках. Вместо попыток предугадать проблемы в будущем, вы решаете только сегодняшние проблемы. Почему? Страхи, которые вы питаете по поводу будущего, как правило, не оправдываются. А потому не толкайте себя в болото, пытаясь бороться с призраками.

С самого начала, мы проектировали наши продукты в соответствии с концепцией меньшего объема кода. При малейшей возможности мы разделяем сложные задачи на простые. Мы нашли решения простых задач, которые легче не только воплощать или поддерживать, но и понимать, и использовать. Все это – часть того, как мы отличаемся от конкурентов; вместо того, чтобы разрабатывать продукты, которые делают больше, мы разрабатываем те, которые делают меньше.

- Меньшим объемом программы легче управлять.
- Меньший объем программы – меньше кода, а это значит
- Меньше скучной работы по сопровождению (и более счастливый персонал).
- Меньший объем программы снижает стоимость изменений – так что вы можете быстрее адаптироваться к обстоятельствам. Вы можете менять свои решения без того, чтобы менять вагоны кода.
- Меньше кода – меньше ошибок.
- Меньше кода – меньше техподдержки.

Какие функции оставить, а какие исключить – тут решение тоже связано с уменьшением объема программы. Не бойтесь отказать в выполнении запроса, который слишком труден. Если функция не является абсолютно критичной – вы сэкономите время и усилия, уменьшите путаницу тем, что откажетесь от нее.

Тише едешь – дальше будешь. Появилась идея – подождите неделю, прежде чем ее воплощать, посмотрите, будет ли она казаться такой же хорошей, когда шум спадет. Помаринуйте идею – и, может быть, за это время вам в голову придет еще более простое решение.

Поощряйте контрпредложения от программистов.
Вот что хорошо было бы от них слышать: “Если делать это как вы предлагаете - на это уйдет 12 часов. Но я могу сделать это за час. В таком случае программа будет делать x и не будет делать y.” Почувствуйте, как программа сопротивляется добавлению лишних функций. Научите программистов отстаивать свою точку зрения на то, как лучше написать программу.

Также ищите обходные пути вокруг необходимости написания большего количества кода. Можете ли вы изменить экран так, что он предложит клиентам альтернативный путь – без того, чтобы менять модель программы? Например, можно ли предложить пользователям загружать картинки только определенного размера – без того, чтобы производить обработку изображений на сервере?

Для каждой функции, которая попадает в вашу программу, спрашивайте себя: а нет ли другого способа ее добавить, используя меньшее количество кода? Пишите только тот код,

который вам нужен, и не более того. Ваше приложение будет более поджарым – и более здоровым.

Нет кода более гибкого, чем отсутствие кода!

“Секрет” хорошего программирования совсем не в том, что именно воплотить в коде – а в том, что оставить за его рамками. В том, чтобы определить, где сильные и слабые места программы, и решить, где нужно просто оставить свободное место, вместо того чтобы заполнять его функциональностью.

—Брэд Эпплтон (Brad Appleton), инженер-программист
(из There is No CODE that is more flexible than NO Code!)

Сложность растет не пропорционально размеру

Самое важное правило разработки программного обеспечения – еще и наименее известное. Сложность программы растет не пропорционально ее размеру... И программа из 2000 строк потребует не в два раза больше времени, а гораздо больше, чем программа в половину этого размера.

—The Ganssle Group (из Keep It Small)

Оптимизируйте для счастья

Выбирайте инструменты, которые заинтересовывают и стимулируют вашу команду

Счастливый программист – продуктивный программист. Вот почему мы оптимизируем удовлетворенность работой, и вам тоже стоит это делать. Выбирайте инструменты, базируясь не только на стандартах отрасли или производительности. Смотрите на неосязаемые факторы: чувствуется ли в инструменте страсть, гордость и мастерство? Будете ли вы по-настоящему счастливы, работая в этой среде восемь часов в день?

Это особенно важно при выборе языка программирования. Вопреки общему мнению, языки программирования не равноценны. В то время как на любом языке можно написать практически любую программу, хороший язык сделает вашу задачу не просто возможной и терпимой, а радостной и дающей силы. Мы о том, что маленькие детали ежедневной работы должны приносить радость.

Счастье заразительно. Счастливые программисты поступают правильно. Они пишут простой, читабельный код. Они придерживаются ясных, выразительных, читабельных подходов. Им нравится то, что они делают.

Мы нашли наше программистское счастье в языке Ruby – и мы поделились им с другими разработчиками через нашу платформу Rails. И Ruby, и Rails проповедуют идею оптимизации для людей и их счастья. Мы приглашаем вас опробовать эту комбинацию.

В общем, вашей команде нужно использовать инструменты, которые станут любимыми. Мы говорили тут о языках программирования, но то же верно для программ, платформ и всего остального. Выберите инструменты, которые не оставят разработчиков равнодушными. Вы получите мотивацию, и, как следствие, лучший продукт.

Какие инженеры вам нужны

Причина того, что я хотел создавать наше приложение на Ruby on Rails, в том, что эта платформа элегантна, продуктивна и красиво спроектирована. Она привлекает инженеров, котоые заботятся именно об этом... а это именно такого рода инженеры, какие вам нужны, потому что они создают именно такие красивые, элегантные и продуктивные программы, которые вам нужны, чтобы завоевать рынок.

—Чарльз Джолли (Charles Jolley), Управляющий Директор, Nisus Software (из Signal vs. Noise)

Говорит код

Слушайте, когда ваш код сопротивляется

Прислушивайтесь к своему коду. Он будет высказывать предложения. Он будет сопротивляться. Он расскажет, где стоят ловушки. Он предложит новые пути решений. Он поможет вам держаться модели меньшего объема программы.

Что, добавление новой функции требует недель времени и тысяч строк кода? Это код говорит вам, что, возможно, существует более легкий способ. Нашли более простое решение, которое можно воплотить за час вместо десяти? Это код вам подсказывает. Прислушайтесь.

Ваш код подскажет вам решения, которые дешевы и легки. Замечайте, когда появляется более простой путь. Разумеется, функция, которую легко сделать, может быть не в точности такой, какую вы себе представляли вначале – ну и что? Если она работает достаточно хорошо и оставляет вам больше времени на другие дела – оставьте ее.

Послушайте

Не беспокойтесь о дизайне; если вы прислушиваетесь к своему коду, хороший дизайн появится сам... Прислушивайтесь к техническому персоналу. Если сотрудники жалуются на то, как трудно вносить изменения – отнеситесь к этому серьезно и дайте им достаточное количество времени на исправления.

—Мартин Фаулер (Martin Fowler), Chief Scientist, ThoughtWorks (из Is Design Dead?)

Если бы программистам платили за то, чтобы убирать код...

Если бы программистам платили за то, чтобы убирать код из программы вместо того, чтобы добавлять его, программы были бы намного лучше.

—Николас Негропонте (Nicholas Negroponte), профессор медиа-технологий, MIT (из And, the rest of the (AIGA Conference) story)

Разберитесь с долгами

Расплачивайтесь по долгам вашего кода и дизайна

Мы обычно говорим о долгах в денежном выражении, но долги могут принимать и другие

формы. Вы Сс;ожете быстро обрасти долгами кода и дизайна.

Наваяли блок кода, который функционален, но все еще неопрятен – вот вы и набрали долгов. Набросали дизайн по принципу “и так сойдет” – ваши долги выросли опять.

Время от времени так поступать можно. Часто такая техника помогает поскорее довести проект в стиле Get Real до конца. Но все равно нужно признать эти долги и рано или поздно расплатиться с ними – вычистить неопрятный код, переделать ту страницу, которая была сделана так себе.

Точно так же, как вы откладываете часть дохода, чтобы заплатить налоги, регулярно выделяйте время на то, чтобы расплатиться с долгами по коду и дизайну. Если этого не делать, придется платить проценты (исправлять кривой код), вместо того, чтобы платить основную сумму (и двигаться вперед).

Открытые двери

Выпустите данные в мир через RSS, API и т.п.

Не пытайтесь запереть ваших клиентов. Позвольте им получить принадлежащую им информацию когда они хотят и как они хотят. Для этого вам придется отказаться от идеи запечатать данные. Выпустите их. Дайте людям доступ к их информации через RSS. Обеспечьте API, позволяющие другим разработчикам подсоединяться к вашей программе. Поступая так, вы облегчаете жизнь клиентам и расширяете возможности своей программы.

Раньше к RSS-фидам относились как к хорошему способу следить за обновлениями блогов и новостных сайтов. Но у них есть и другие сильные стороны. Они также позволяют клиентам быть в курсе изменяющегося контента приложения без того, чтобы постоянно в него входить. С фидами Basecamp пользователи могут ввести адрес в новостную ленту и быть в курсе того, что делается с проектом: сообщения, списки задач, этапы - без того, чтобы постоянно проверять все это на сайте.

API позволяют разработчикам создавать добавки к вашим продуктам – которые, в свою очередь, могут оказаться неоценимыми. Например, Backpack предоставляет API, которыми Chipt Productions воспользовался для построения программы Mac os x Dashboard. Программа позволяет добавлять и редактировать напоминания, списки и другое на рабочем столе. Клиенты с большим одобрением оценили эту программу, многие даже сказали, что благодаря ей они стали пользователями Backpack.

Вот еще примеры компаний, выпускающих данные наружу и получающих в результате эффект бумеранга:

- **Google Maps API** положило начало интересной тенденции смешанных сайтов, когда люди получают информацию из других источников (например, объявления о квартирах) и наносят их на карту.
- Linkrolls предлагает клиентам способ получить новейшие закладки del.icio.us, показанные на их сайтах.
- **Flickr** предоставляет другим продавцам доступ к своим коммерческим API, так что клиенты могут покупать фотоальбомы, запасные хранилища данных для dvd, плакаты и марки. “Цель – открыть все по максимуму и дать вам наибольшее разнообразие выбора того, что вы можете сделать со своими фотографиями”, - говорит Стюарт Баттерфилд из Flickr.

Программка, которая меняет дело

Когда компания 37signals выпустила Backpack, мое первое впечатление было...э-э.

Так что, когда Chipt Productions выпустили программку для Backpack для Tiger – и невозможно было ее не попробовать – я посмотрел на Backpack во второй раз. Результат? Совсем другое дело.

Сейчас, когда мне приходит в голову новая идея, я запускаю программку, печатаю, отправляю сообщение - готово. По электронной почте приходит нечто, что я хочу посмотреть? Запускаю программку, печатаю, отправляю – готово. Эта программка установлена – на каждом компьютере Mac, который я использую. И благодаря тому, что программка сетевая – не нужно заботиться о контроле версий или синхронизации – можно просто вводить информацию, не задумываясь, куда она пойдет или как получить к ней доступ позднее.

—Тодд Домини (Todd Dominey), основатель компании Dominey Design
(из Trying on Backpack)

Слова глава 11

В функциональной спецификации нет ни грамма функциональности

Не составляйте функциональных спецификаций

Эти черновые документы обычно не имеют почти ничего общего с конечным продуктом. И вот почему:

Функциональные спецификации - это фантазии

Они не отражают реальности. Приложение не является реальностью до тех пор, пока строители не начнут его строить, дизайнеры – оформлять, люди – использовать. Спецификации – это только слова на бумаге.

Функциональные спецификации как средство умиротворения

Они для того, чтобы каждый почувствовал себя вовлеченным и счастливым. Приятное чувство, но не такое уж полезное. Спецификации никргда не говорят о принятии неприятных решений и не открывают истинных затрат – а это как раз то, что нужно для построения хорошего приложения.

Функциональные спецификации создают лишь иллюзию соглашения

Какое-то количество людей, согласных с текстом – это не есть соглашение. Каждый может читать тот же самый текст и думать о разном. Это безусловно всплывет позже, когда станут говорить: “Обождите, это не то, что я подумал”, “Что? Да это же не так, как мы описали”, “Да, это именно так, и мы все согласились с этим, тут даже есть ваша подпись”. Вы знаете,

как это обычно бывает.

Функциональные спецификации заставляют вас принимать наиболее важные решения именно тогда, когда у вас меньше всего информации

Вы меньше всего знаете о приложении, когда вы начинаете его строить. Чем больше вы его строите, чем больше используете – тем больше вы его знаете. Вот когда вам стоит принимать решения – когда у вас больше информации, а не когда меньше.

Функциональные спецификации ведут к перегруженности функциями

В стадии спецификаций вас ничто не ограничивает. Ничего не стоит просто писать и добавлять все новые и новые пункты. Вы можете уступить кому-то надоедливому тем, что добавите его любимую функцию. Потом вы будете разрабатывать с тем, чтобы удовлетворить эти пункты, а не людей. Именно так получаются перегруженные сайты с 30 кнопками по верху экрана.

Функциональные спецификации не дают вам развиваться, меняться и пересматривать

Вот функция программы подписана и согласована. Даже если вы поймете в процессе разработки, что эта идея была плохой, вы на ней застряли. Спецификации нет дела до действительности, когда как только вы начинаете что-то строить, все меняется.

Так что же вы должны делать вместо написания спецификации? Найдите более короткую альтернативу – такую, которая продвигает вас по пути создания. Напишите рассказ в одну страницу о том, что именно приложение должно делать. Используйте простой язык и сделайте это быстро. Если вам потребуется более одной страницы, значит, вы очень усложняете. Этот процесс не должен занять более одного дня.

Потом начните строить интерфейс – он и будет альтернативой функциональной спецификации. Нарисуйте несколько эскизов на бумаге. Потом начните кодировать это в html. В отличие от текста, который могут понять по-разному, дизайн интерфейса будет представлять то общее, с чем все могут согласиться.

Двусмысленность уходит, когда все начинают видеть на экране одно и то же. Постройте интерфейс, на который каждый может смотреть, пользоваться им, кликать мышкой, почувствовать его – до того, как начнете беспокоиться о внутреннем коде. Встаньте на передний край пользовательского опыта, насколько это возможно.

Забудьте о подписанных раз и навсегда спецификациях. Они заставляют вас принимать крупные, ключевые решения слишком рано в процессе разработки. Обойдите стороной стадию спецификации, и вам удастся быть гибкими и сделать изменения дешевле.

Бесполезные спецификации

Спецификация по большей части бесполезна. Я никогда не видел спецификации настолько большой, чтобы быть и одновременно и полезной, и точной.

В то же время я встречал много раз полную туфту, которая была основана на спецификациях. Это наиболее плохой способ писать программы, так как он по определению значит, что программа писалась, чтобы соответствовать теории, а не действительности.

—Линус Торвальдс (Linus Torvalds), создатель Linux (из: Linux: Linus On Specifications)

Боритесь с создателями препятствий

Я обнаружил, что люди, настаивающие на подробных описаниях требований к программе до того, как начать разработку, в действительности являются создателями препятсятвий, пытающимися замедлить процесс (и которым обычно нечего сказать по поводу дизайна или инновационного мышления).

Все наши лучшие работы начинались с небольшого количества идей о том, как улучшить сайт, быстрого создания прототипа (статического), небольшого изменения дизайна и затем построения реального прототипа с реальными данными. После этого настоящий проект быстро набирал обороты и выполнялся с хорошим результатом.

—Марк Галлахер (Mark Gallagher), разработчик корпоративных интранет-сайтов (из Signal vs. Noise)

Не рождайте мертвых документов

Уберите ненужное бумаготворчество

Избегать функциональных спецификаций хорошо, но этого мало. Предотвращайте ненужное бумаготворчество везде, где можете. Если только документ не собирается воплотиться во что-то реальное – не создавайте его.

Стройте, а не пишите. Если вам нужно что-то объяснить, попробуйте это изобразить и сделать в качестве прототипа, вместо того чтобы долго документировать. Реальный интерфейс или прототип уже находятся на пути воплощения в реальный продукт. А лист бумаги, напротив, – лишь на пути в мусорную корзину.

Вот пример. Если документу-каркасу предназначено застыть и никогда не перерасти в настоящий продукт – не тратьте на него время. А если, начав с каркаса, вы строите на его базе настоящий проект, тогда да, начните с каркаса.

Документы, живущие отдельной от приложения жизнью, ничего не стоят. Они не ведут никуда. Все, что вы делаете, должно воплощаться в реальность.Если документ останавливается до того, как воплотиться в реальность, – он мертв.

Никто не будет это читать

Я не могу сосчитать, как много многостраничных спецификаций лежало мертвым грузом, непрочитанными, собирало пыль, тогда как мы кодировали, обсуждали проблемы, задавали вопросы и тестировали. Приходилось даже иметь дело с разработчиками, которые потратили часы на написание длинных посланий по электронной почте или документов о стандартах программирования – которые тоже никто не читал.

Сетевые приложения не растут благодаря документации. Разработка программ – постоянно меняющийся, итеративный процесс, который включает в себя взаимодействие, принятие мгновенных решений и непредвиденные темы, которые приходят в процессе. Ничего из этого не представляется возможным (либо нужным) закрепить на бумаге заранее.

Не теряйте время на то, чтобы заполнить текстом длинные тома; никто не будет их читать. Утешайте себя тем, что если продукту оставить достаточно места для самостоятельного роста, он вырастет так, что в любом случае не будет похож на то, что вы про него напишете.

—Джина Трапани (Gina Trapani), веб-разработчик и редактор Lifehacker, the productivity and software guide

---

Расскажите короткую историю

Пишите рассказы, а не описывайте детали

Если вам нужны слова, чтобы описать новую функцию или изложить идею – напишите об этом короткий рассказ. Не углубляйтесь в технические или архитектурные детали, просто расскажите историю. Сделайте это на человеческом языке, как бы вы это сделали в разговоре.

Не нужно делать это литературным произведением. Просто расскажите, в какой последовательности все происходит. Если вы можете включить этот рассказ в контекст интерфейса программы, тем лучше.

Держитесь опыта, а не зацикливайтесь на деталях. Думайте о стратегии, а не о тактике. Тактика определится, когда вы начнете строить данную часть программы. Пока же вы просто хотите пол43;чить рассказ, который станет началом разговора и приведет вас в нужную колею.

Пользуйтесь обычными словами

Вставьте настоящий текст вместо lorem ipsum

Слова lorem ipsum dolor – признанные друзья дизайнеров. Текст-пустышка помогает понять, как будет выглядеть дизайн, когда он будет воплощен. Но текст-пустышка может быть опасным.

Lorem ipsum меняет ваш взгляд на программу. Он сводит текстовое содержание к элементу визуального дизайна – форме текста – вместо того, чем он должен быть: значимой информацией, которую кто-то должен будет вводить и/или читать. Текст-пустышка означает, что вам не удастся увидеть тех необходимых вариаций, которые безусловно появятся, когда будет введена настоящая информация. Это значит, что вы не почувствуете, как это – заполнять поля на вашем сайте. Текст-пустышка – это завеса между вами и действительностью.

Вам нужна настоящая копия, чтобы знать, какой длины должны быть поля. Вам нужна настоящая копия, чтобы видеть, как таблицы будут расширяться или сжиматься. Вам нужна настоящая копия, чтобы знать, как в действительности выглядит ваше приложение.

Как можно скорее используйте настоящие и подходящие слова. Если ваш сайт или приложение требует ввода данных, вводите реальные данные. И, собственно, печатайте текст – а не просто переносите его из другого источника. Если это имя, вводите реальное имя. Если город, вводите название существующего города. Если это пароль, ввод которого нужно повторить, введите его дважды.

Конечно, намного легче просто пробежать формы сверху вниз и заполнить поля мусором (“asdsadklja” “123usadfjasld” “snaxn2q9e7”), чтобы разобраться с ними побыстрее. Но это – неправда. Это не то, что придется делать вашим клиентам. Хорошо ли идти по короткой

дороге, а клиентов отправлять по длинной? Когда вы вводите в пожарном порядке, вы не знаете, как в действительности выглядит заполнение этой формы.

Делайте, как ваши клиенты – и вы станете их лучше понимать. Когда вы лучше их понимаете и чувствуете то, что чувствуют они, вы построите лучший интерфейс.

Мусор Lorem Ipsum

Когда вы не заставляете воображение представлять, каким “может быть” содержание, рассмотрение дизайна потеряно. Значение спрятано, потому что “это всего лишь текст”, понимабельность теряется, потому что никто не задумывался, что эти тексты предназначены для чтения. Возможности остаются нереализованными, потому что тот мусор lorem ipsum, который вы использовали, не предлагает возможностей. А потом текст делается маленьким-маленьким, потому что, если он не нужен, мы можем создать много таких симпатичных белых пятен.

—Том Смит (Tom Smith), дизайнер и разработчик (из I hate Lorem Ipsum and Lorem Ipsum Users)

Очеловечьте ваш продукт

Какой у вашего продукта тип личности?

Подумайте о своем продукте как о человеке. Каким человеком вы бы хотели его видеть? Вежливым? Неумолимым? Прощающим? Требовательным? Веселым? Бесстрастным? Серьезным? Разболтанным? Хотите, чтобы он выглядел параноидальным или доверяющим? Всезнайкой? Или скромным и обаятельным?

Когда вы примете решение, всегда имейте в виду эти черты, пока строите продукт. Пусть они помогут вам в принятии решений по поводу копирайта, интерфейса и функциональности. Всякий раз, когда вносите изменение, спросите себя, насколько это изменение соответствует типу вашего приложения.

У вашего продукта есть голос – и он разговаривает с вашими клиентами 24 часа в сутки.

Цена и регистрация глава 12

Бесплатные образцы

Раздавайте что-нибудь бесплатно

Мир полон шума и суеты. Чтобы вас заметили среди всего этого, раздавайте что-нибудь бесплатно.

Умные компании знают, что раздача подарков – замечательный способ завоевать благосклонность клиентов. Посмотрите на компанию Apple. Они предлагают программу iTunes бесплатно, с тем чтобы создать спрос на iPod и на музыкальный магазин iTunes. В реальном мире, магазины делают то же самое. Кофейная компания Starbucks утверждает, что

каждые пять напитков, розданных бесплатно, влекут за собой одну новую покупку. А что, весьма неплохо.

В нашем случае, Writeboard и Ta-da list – полностью бесплатные программы, с помощью которых мы приводим клиентов к использованию других наших продуктов. Также мы всегда предлагаем некоторую бесплатную версию для всех наших программ.

Мы хотим, чтобы люди могли почувствовать продукт, интерфейс и полезность того, что мы создали. Когда они уже на крючке, намного более вероятно, что они захотят перейти на какую-нибудь из платных версий (что позволит иметь большее количество проектов или страниц, а также даст доступ к дополнительным функциям, таким как загрузка файлов и ssl-шифрование данных).

Миниатюрные порции

Предлагайте миниатюрные порции. Заготовьте специализированные, уменьшенные предложения, пусть клиенты их пробуют. Примите за правило делить хотя бы один продукт или услугу на маленькие кусочки, которые недороги, легки или интересны.

—Бен МакКоннелл (Ben McConnell) и Джеки Хуба (Jackie Huba), авторы Church of the Customer Blog
(из What is customer evangelism?)

Подарите одну из ваших лучших песен

Подумайте о том, чтобы распространять одну из ваших песен (одну на альбом) бесплатно – как рекламный ролик фильма – как хит, посылаемый на радио – песню, которая побудит людей покупать вашу музыку.

Не беспокойтесь о пиратстве в отношении этой песни. Пусть ее слушают, переписывают, посылают друзьям, раздают. Будьте уверены, что если ее послушают, то будут готовы заплатить за другие.

—Дерек Сильверс (Derek Sivers), президент и программист компаний CD Baby и HostBaby (из Free Promo Track)

Легко войти, легко выйти

Сделайте регистрацию и отказ безболезненными

Сделайте начало использования вашей программы – и окончание использования – как можно проще.

Если я – клиент, который хочет использовать вашу программу, процесс регистрации должен быть безболезненным, на уровне “для чайников”. Поставьте большую и ясную кнопку регистрации на каждой странице вашего маркетингового сайта. Расскажите, как все просто: ”От регистрации до использования – всего за 1 минуту!”

Всегда нужно обеспечить бесплатную опцию, чтобы пользователи могли увидеть демо-версию без того, чтобы вводить данные кредитной карточки. Некоторые из наших конкурентов требуют звонка, договоренности или специального пароля, чтобы получить демо-версию. А зачем? Мы даем попробовать наши программы всем бесплатно в любое время.

Сделайте форму для регистрации как можно короче. Не запрашивайте информацию, которая

вам не нужна, и не заваливайте пользователей длинными анкетами.

То же соблюдайте и для процесса отмены. Вы не хотите замыкать людей внутри вашего проекта. Нам не очень приятно, когда люди решают отказаться от пользования Basecamp, но мы никогда не делаем этого процесса унизительным либо запутанным. “Закрыть мой аккаунт” - это ссылка, ясная как день, на странице управления профилем пользователя. Не должно быть ни обязательного уведомления по электронной почте, ни формы, обязательной для заполнения, ни каких-либо вопросов.

Также создайте условия, чтобы люди могут получить свои данные, если они решат уйти. Мы обеспечиваем пользователям возможность легко экспортировать их сообщения и комментарии в формате xml в любое время. Это их данные, поэтому они могут делать с ними что хотят.

Это критический фактор, потому что давая людям власть над их информацией, вы создаете доверие. Вы обеспечиваете им мост к их островкам данных. Вы позволяете им уйти без потерь, если они найдут лучшее предложение. Это – правильно, и это строит добрую волю.

Уйти с легкостью

Не удерживайте пользователей вопреки их воле. Если они хотят уйти, позвольте им забрать все, что они создали, будучи на вашем сайте, и уйти... без штрафов... Вы должны держать двери вашего сарая открытым и сосредоточиться на кормлении клиентов, с тем чтобы они возвращались, потому что хотят этого, а не потому, что застряли в дверях.

—Чарли О’Доннелл (Charlie O'Donnell), аналитик компании Union Square Ventures (из 10 Steps to a Hugely Successful Web 2.0 Company)

Все эти "Растишки", зайчишки и прочие завлекушки с наклейками и раскрасками — уловки для детей

Избегайте долгосрочных контрактов, платы за подключение и т.д.

Никто не любит долгосрочных контрактов, штрафов за раннее прерывание контракта, платы за подключение. Значит, избегайте всего этого. Наши продукты оплачиваются помесячно. Нет никаких контрактов, и отказаться от пользования вы можете в любой момент. И платы за подключение тоже нет.

Не придумывайте всякие "фокусы" чтобы получить побольше денег. Зарабатывайте их.

Подсластите пилюлю

Подсластите пилюлю плохих новостей предварительным сообщением и особыми условиями для ”дедушек”

Вам приходится объявлять плохие новости, такие как увеличение цены? Сделайте это как можно более безболезненным, сообщив об этом заранее. Также подумайте о том, чтобы освободить существующих пользователей от повышения цены на какое-то время. Эти люди обеспечивают вам хлеб с маслом, и вы хотите, чтобы они чувствовали себя уважаемыми, а не жертвами вымогательства.

Продвижение глава 13

Выпуск в голливудском стиле

Анонс – рекламный показ отрывков - выпуск

Если выпустить программу в лесу, где ее некому использовать, произведет ли она фурор? Мы о том, что если выпустить программу без предварительной подготовки, то, скорее всего, никто о ней не узнает.

Чтобы спровоцировать разговоры и ожидание, выпускайте программу в стиле Голливуда: 1) Анонс (teaser, буквально: дразнилка – прим. перев.), 2)Рекламный показ отрывков, 3) Выпуск.

Анонс

За несколько месяцев начинайте намекать. Расскажите людям, над чем вы работаете. Покажите логотип. В своем блоге сообщайте о разработке. Пишите неопределенно, но зароните зерно. Также заведите сайт, где будете собирать адреса электронной почты от интересующейся публики.

На этой стадии также начните окучивать знатоков и экспертов. Это те, кто находится на переднем крае. Они определяют вкусы остальных. Взывайте к их тщеславию и статусу первопроходцев. Пообещайте предоставить им эксклюзивное право предварительного просмотра программы. Если ссылка на вашу программу попадет на сайт типа Boing Boing, Slashdot, или Digg – вы получите много траффика и пользователей. Ваш ранг на Google тоже вырастет.

Предварительный показ

За несколько недель до выпуска начните предварительный показ некоторых возможностей. Дайте пользователям доступ из-за кулис. Опишите тему продукта. Для сайта Basecamp мы поместили скриншоты, выделили напоминалки, этапы, а также другие свойства.

Также расскажите об идеях и принципах, заложенных в программу. Перед выпуском программы Backpack мы опубликовали наш манифест. Поэтому люди стали думать о нашей программе и обсуждать ее.

Вы также можете предложить какому-то количеству пользователей особые “золотые

контрамарки”, дающие им право использовать приложение раньше других. Вы получите бета-тестеров, а они почувствуют то сияние, которое получают люди, являющиеся ранними последователями.

И опять же, поощряйте регистрацию, чтобы у вас было много адресов электронной почты, по которым вы пошлете сообщение о выпуске. К тому моменту, когда мы выпускаем наши приложения, у нас есть D2;ысячи адресов – что является большим подспорьем для набирания оборотов.

Выпуск

Итак, настало время выпуска. Сейчас пользователи наконец-то могут “пойти в кино” и посмотреть на ваше приложение. Отправьте сообщение по электронной почте всем, кто зарегистрировался. Запустите полновесный маркетинговый сайт. Распространяйте проповедь как только можно. Пусть на вас ссылаются блоги. Сообщайте о своем прогрессе. Сколько у вас зарегистрированных пользователей? Сколько обновлений/поправок вы сделали? Покажите движение вперед и поддерживайте его.

На пути к выпуску

Как только мы поняли, что программа Blinksale действительно будет выпущена, мы начали рассылать анонсы участникам нашего списка рассылок. Это те, кто запрашивал информацию о наших проектах. Наши фанаты, если угодно. Если вы уже можете обращаться к группе людей, начните с обращения именно к ним.

Втоое, что мы сделали – получили разрешение рассказать о продукте большему количеству людей. Примерно за шесть недель до выпуска Blinksale мы поместили на нашем вебсайте анонс, в котором объявили, что существует более легкий способ посылать инвойсы в онлайне. Эта страница содержала как раз столько информации, чтобы заронить волнение и неизвестность - без того, чтобы сообщать детали, которые пока должны были остаться в тайне. На странице была форма подписки на новостную рассылку, не требующая ничего, кроме адреса электронной почты (все просто!) – так что все заинтересовавшиеся программой получали возможность узнать о ее выпуске.

Мы замолвили словечко десятку-другому друзей и коллег, которым, как мы считали, продукт тоже мог бы быть интересен, и они начали распространять информацию о странице с анонсом через свои веблоги и сайты. Через несколько дней в нашем списке рассылки были тысячи адресов. Эти люди были очень важны для нас – те, кто хотел побольше узнать о нашем продукте и когда он будет выпущен.

Наконец, примерно за две недели до выпуска, мы пригласили группу друзей, коллег и промышленных магнатов помочь нам в бета-тестировании Blinksale. Это позволило нам показать продукт тем, кто получит выгоду от его использования и поможет нам распространить информацию о продукте после его выпуска. Важно заметить, что мы не требовали ни от кого использовать наш продукт или писать о нем. Мы просто хотели, чтобы продукт увидели и говорили о нем, когда он будет выпущен. В конце концов, если вы хотите подготовить сенсацию таким способом, будьте уверены, что продукт выполняет свои обещания. Иначе это как облака без дождя.

Когда пришел день выпуска, мы разослали сообщение участникам нашего списка рассылки, сообщили нашим друзьям-блоггерам, а также предложили нашим бета-тестерам рассказать о своих впечатлениях. К нашему огромному удовольствию, наши усилия окупились. Сразу после выпуска десятки тысяч пользователей посетили наш сайт, и тысячи из них зарегистрировались для использования продукта.

—Джош Вильямс (Josh Williams), основатель компании Blinksale

Мощный сайт для продвижения

Анонс, предварительный просмотр, выпуск

Лучшее средство для продвижения продукта – сам продукт. Если вы создали приложение, которое действительно нужно клиентам, об этом непременно узнают.

Тем не менее, вам все равно нужен мощный сайт для продвижения. Что туда можно поместить? Вот некоторые идеи.

- **Обзор**: Расскажите о приложении и его достоинствах.
- **Экскурсия**: Покажите пользователям различные свойства.
- **Скриншоты и видеоролики**: Покажите, как выглядит программа и как ею пользоваться.
- **Манифест**: Объясните философию программы и идеи, лежащие в ее основе.
- **Примеры**: Приведите примеры из жизни, иллюстрирующие возможности программы.
- **Отзывы**: Отзывы пользователей, обзоры, пресса и т.д.
- **Форум**: Предложите место, где члены сообщества пользователей смогут помогать друг другу.
- **Цены и регистрация**: Заполучите пользователей как можно быстрее.
- **Блог**: Блоги освежают ваш сайт: помещайте туда новости, советы и т.д.

На волне блогов

Блоги могут быть зачастую эффективнее, чем реклама (и, кстати, они невообразимо дешевле)

Реклама стоит дорого. А оценка эффективности различных типов рекламы может оказаться дороже самой рекламы. Когда у вас нет времени или денег, чтобы пойти по традиционному рекламному пути, подумайте о другом пути: рекламируйте через блоги.

Начните с создания блога, который не только хвалит ваш продукт, но еще и предлагает полезные советы, решения, ссылки и т.д. На наш блог Signal vs. Noise приходят тысячи читателей в неделю, благодаря полезным, информативным и интересным фрагментам и историям, которые мы помещаем ежедневно.

Таким образом, когда пришло время продвигать наш первый продукт, Basecamp, мы с этого и начали. Мы сообщили в Signal vs. Noise - и информация начала распространяться. Jason Kottke, the BoingBoingers, Jim Coudal и многие другие хозяева популярных блогов помогли увеличению видимости продукта, и дело пошло.

Ta-da Lists – еще один замечательный пример маркетинга через блоги. Мы предварили выпуск Ta-da одним-единственным сообщением в Signal vs. Noise, и через несколько недель о продукте упоминалось более чем в 200 блогах, и более 12000 людей зарегистрировались для использования продукта. В случае Backpack новость распространилась еще быстрее. Через 24 часа после выпуска были зарегистрированы более 10000 пользователей.

Начинайте рекламировать как можно раньше

Распространите слухи и получите подписчиков как можно скорее

Мы уже говорили об этом, но повторим: заведите сайт и начните собирать адреса электронной почты как можно раньше. Выберите домен, повесьте логотип и, может быть, одно-два предложения, описывающие, что будет делать ваше приложение. И пусть вам оставляют свои адреса. Теперь вы на пути к основанию сообщества пользователей, подготовленных к выпуску продукта, и ожидающих его.

Продвижение через обучение

Поделитесь знаниями с миром

Когда в программе Jeopardy играет учитель, ведущий Алекс Требек часто говорит, что это благородная профессия. И он прав. Есть нечто замечательное и благодарное в том, чтобы делиться знаниями с другими. А когда предмет обучения – ваша же программа, вы достигаете двух целей: делитесь знаниями с обществом и получаете аудиторию.

Как технология продвижения, обучение – мягкий путь к тому, чтобы ваше имя – равно как и название вашего продукта – появилось перед множеством людей. И вместо того, чтобы заниматься навязыванием продукта, вы получаете внимание за предоставление ценной услуги. Это создает позитивный настрой, недостижимый для традиционного маркетинга. Те, кого вы научите, потом станут вашими проповедниками.

Обучение может принимать различные формы. На вашем сайте помещайте советы и решения, которыми читателям захочется поделиться с другими. Выступайте на конференциях и оставайтесь после выступления, чтобы поговорить с участниками. Проводите семинары, чтобы любопытные поклонники имели возможность узнать больше и поговорить с вами вживую. Давайте интервью различным изданиям. Пишите статьи. И книги тоже. ;)

Пример из нашего опыта – the Yellow Fade Technique, метод, который мы придумали, чтобы выделить недавно измененную область страницы. Мы поместили сообщение об этом в Signal vs. Noise. Это сообщение передавалось из уст в уста и имело тысячи и тысячи просмотров (и до сих пор у него огромный трафик).

Это сообщение сработало и на уровне образования, и на уровне продвижения. Урок был усвоен, и многие из тех, кто никогда бы не узнал о наших продуктах, получили возможность с ними ознакомиться. Еще пример: когда мы создавали Ruby on Rails, мы решили сделать код открытым. Это оказалось мудрым шагом, потому что мы поделились с сообществом, снискали признание для нашей команды, получили полезную обратную связь и стали получать патчи от программистов со всего мира.

Преподавание улучшает вашу карму. Вы платите вперед. Вы помогаете другим. И вы получаете продвижение. И даже порой признание. Вот вы - что вы знаете такого, о чем мир хотел бы услышать?

Платите вперед

Раздел статей и советов в нашем блоге – самое популярное место на нашем сайте. Делясь знаниями о маркетинге через электронную почту с нашими пользователями, мы помогаем им извлечь максимум возможностей из наших программ. Если они смогут лучше обслуживать своих клиентов, их оборот будет больше, а значит, и наш оборот тоже – все выигрывают.

Свободное распространение наших знаний также помогло нам позиционировать себя как экспертов в нашей области и укрепило наши отношения с существующими клиентами. Они знают, что мы заботимся о качестве их работы. Также, мы получаем огромный траффик через поисковые системы и от блоггеров, которые делятся нашими статьями со своими читателями. Эти люди никогда бы не услышали о наших программах, если бы мы не написали этой статьи.

—Давид Грейнер(David Greiner), основатель компании Campaign Monitor

---

Пища функциональности

Они этого жаждут – дайте же им скорее

Добавление новых или интересных функций – хороший способ усилить разговоры о вашем приложении. Группы по интересам очень любят пережевывать “пищу функциональности” и выплевывать ее обратно в сообщество. Ладно, аналогия не слишком аппетитная, но смысл вы поняли.

Например, мы привлекли тонну внимания сообщества разработчиков к Basecamp тем, что при разработке использовали новую платформу Ruby on Rails.

Элементы Ajax, которые мы использовали в приложениях, тоже снискали много внимания и даже побудили журнал Business 2.0 назвать 37signals “ключевым игроком Ajax” наряду с такими крупными именами, как Google, Yahoo, Microsoft и Amazon.

Еще пример. Блоггеры оценили поддержку RSS в программе Basecamp, потому что это было одним из первых примеров RSS в бизнесе.

Интеграция iCal, казалось бы, небольшая деталь, обеспечила нам внимание множества сайтов, посвященных Mac – которые, вероятно, без нее никогда не упомянули бы о нашей программе.

Небольшие команды идут на шаг вперед, когда дело касается воплощения новых идей в программы. Там, где более крупным компаниям приходится преодолевать бюрократические препоны, вы можете быстро воплотить новые идеи и снискать внимание за их использование.

Проехаться на запятках технологии дня – эффективный и дешевый способ обрести внимание. Мы не говорим: поддерживайте самую новую, еще никому не известную технологию, только чтобы вас заметили. Но если вы используете что-то новое или примечательное – вперед, дайте об этом знать интересующимся группам.

Следите за логами

Изучайте свои логи, чтобы следить за дискуссиями

Вам полезно знать, кто о вас говорит. Проверьте свои логи и узнайте, что откуда берется. Кто на вас ссылается? Кто ругается? Какие из блогов, попавшие в списки на Technorati, Blogdex, Feedster, Del.icio.us и Daypop, говорят о вас?

Узнайте это, а затем дайте другим почувствовать ваше присутствие. Оставьте комментарии в этих блогах. Поблагодарите людей за ссылки. Спросите, хотят ли они, чтобы их включили в списки рассылки о предстоящих выпусках, обновлениях и т.д. Соберите положительные отзывы и создайте страницу дискуссий и благодарностей на вашем сайте. Похвалы пользователей – замечательный способ продвижения вашей программы, потому что отзывы третьих лиц вызывают большее доверие у большинствы людей.

Если отзывы отрицательные, все равно отнеситесь к ним внимательно. Покажите, что вы к ним прислушиваетесь. На критику отвечайте обдуманно. Что-то типа: “Мы ценим ваши отзывы, но мы сделали это таким-то способом, потому что...” Или: “Вы затронули важную проблему, и мы над ней работаем”. Вы смягчите критику и покажете человеческое лицо вашего продукта. Удивительно, насколько обдуманн4b;й комментарий в блоге может рассеять напраслину и даже превратить жалобщиков в сторонников.

Продажи в процессе

Предлагайте возможности апгрейда в самом приложении

Каждый знает, как предлагать на маркетинговом сайте. Но продажи не останавливаются на этом. Если у вас зонный ценовой план (или бесплатная версия вашей программы), не забудьте сообщать о возможности апгрейда в самой программе.

Расскажите, что вы уберете барьеры, если они перейдут на следующий уровень. Например, в Basecamp вы не можете загружать файлы, если у вас бесплатная версия программы. Когда кто-то пытается загрузить файл, мы не просто запрещаем это. Мы объясняем, почему опция недоступна, предлагаем перейти на платную версию и объясняем, почему это стоит сделать. Такой же подход используется, когда мы предлагаем существующим пользователям перейти на обслуживание более высокого уровня, когда они выходят за рамки своего текущего плана.

Существующие клиенты – ваш наиболее перспективный рынок для дальнейших продаж. Не стесняйтесь развивать бизнес с теми, кто уже знает и использует ваш продукт.

На крючке названия

Дайте вашему приложению легко запоминающееся название

Многие совершают большую ошибку, когда считают, что название продукта должно быть

сверхинформативным. Не стоит выбирать название, которое содержит детальное описание продукта. Обычно в результате выходит немудреное и легко забывающееся название. Название Basecamp лучше, чем что-то типа Project Management Center или ProjectExpress. Writeboard – лучше, чем CollaborEdit.

Также не слишком увлекайтесь фокус-группами и комитетами в процессе поиска имени. Выберите имя короткое, яркое, запоминающееся – и вперед.

И не переживайте, если не удалось получить доменное имя, которое вы хотели. Проявите фантазию и найдите что-то похожее, добавив пару букв (например, backpackit.com или campfirenow.com).

Полегче

Разве технологическая промышленность не понимает, что придумать яркое, не требующее объяснений имя выгодно ей самой? Они продадут больше, все равно чего, потому что они не напугают потребителей, которые сочли бы, что кучка надменных инженеров не пускает их в свое общество. Технология тоже ухватится за это быстрее. Новый продукт будет легче описывать, использовать и покупать – что для компаний значит - легче продавать.

—Дейвид Пог (David Pogue), обозреватель, New York Times (из What's in a Product Name?)

Поддержка глава 14

Почувствуйте эту боль

Сокрушите стены между разработкой и поддержкой

В ресторанном бизнесе есть большая разница между теми, кто работает на кухне, и теми, кто работает непосредственно с клиентами. Важно, чтобы каждая сторона понимала другую и знала ее интересы. Вот почему во многих кулинарных школах и ресторанах повара часто заменяют официантов, чтобы работники кухни взаимодействовали с клиентами и видели, что в действительности происходит на переднем крае.

Многие компании-разработчики программного обеспечения имеют похожее разделение. Архитекторы и программисты работают на “кухне”, тогда как техподдержка работает с клиентами. К сожалению, это означает, что те, кто “варит и печет” программы, никогда не слышат того, что говорят их клиенты. Это таит в себе проблемы, ибо слышать клиентов – лучший способ узнать реальные достоинства и недостатки вашего продукта.

Решение? Избегайте постройки стен между вашими клиентами и командой разработчиков. **Не передавайте техподдержки телефонному центру или сторонней организации.** Осуществляйте ее сами. Вы, вся ваша команда, должны знать, что говорят ваши клиенты. Когда они недовольны, вы должны знать об этом. Вы должны слышать их отзывы. И вы тоже должны быть недовольны.

В 37signals на всю электронную почту техподдержки отвечают лично те, кто создавал продукт. Почему? Во-первых, потому, что они могут предоставить лучшую поддержку для пользователей. Пользователи получают ответ прямо от тех, кто строил продукт. Также, это приближает нас к людям, которые используют наш продукт, и к их проблемам. Когда они страдают, мы тоже страдаем. Мы можем сказать, что чувствуем их боль – и мы ее действительно чувствуем.

Вам может показаться заманчивым использование статистических результатов для того, чтобы определить возможные источники проблем. Но цифры на бумаге – совсем не то, что голоса. Вам нужно убрать как можно больше преград, стоящих между вами и реальными голосами ваших клиентов.

Передний край – то место, где все происходит. Идите туда. Пусть ваши повара поработают официантами. Читайте электронную почту от клиентов, прислушайтесь к их замечаниям и предложениям, учитесь на них

Уберите посредника

Почти вся разработка, поддержка и маркетинг в компании Campaign Monitor делается силами двух человек. Даже если нам понадобится расширить команду, мы никогда не будем отделять техподдержку от разработки. Отвечая на каждый запрос лично, мы ставим себя на место наших клиентов и видим происходящее с их точки зрения.

Важно понять, почему ваш клиент хочет чего-то, а не только то, что именно он хочет. Очень часто контекст имеет прямое влияние на то, как мы будем реализовывать продукт. Уберите посредника. Намного легче предоставить клиентам то, что они хотят, когда вы их так слышите.

Я обсуждал это со многими людьми, и меня часто спрашивали: “А не проще ли нанять молодого специалиста и передать ему техподдержку?” Поставьте себя на место своего клиента. Если вы хотите, чтоб ваш бифштекс был приготовлен особым образом – с кем бы вам это было бы лучше согласовать: с младшим официантом или с тем поваром, который будет готовить этот бифштекс?

—Давид Грейнер(David Greiner), основатель компании Campaign Monitor

---

Нулевое обучение

Используйте встроенную систему помощи и списки часто встречающихся вопросов, чтобы ваш продукт не требовал справочников или тренингов

Чтобы пользоваться сайтами Yahoo, Google или Amazon, вам не нужны учебники или справочники. Почему бы вам не создать продукт, которому они тоже не нужны? Стремитесь создать продукт, не требующий обучения.

Как это сделать? Как мы упоминали ранее, начните с создания максимально простой программы. Чем меньше ее сложность, тем меньше усилий вы приложите для спасения заблудившихся в ней пользователей. Далее, встроенная система помощи и списки часто встречающихся вопросов - хороший способ помочь пользователям найти ответы на вопросы еще до того, как они решат обратиться к техподдержке.

Например, мы предлагаем именно такую поддержку на странице, позволяющей пользователям загрузить свой логотип в программу Basecamp. Некоторые пользователи теряются, когда видят свой старый логотип из-за настроек кэша браузера. Поэтому рядом с областью загрузки логотипа мы добавили ссылку на список часто встречающихся вопросов, где инструктируем пользователя перегрузить браузер, чтобы увидеть новый логотип. До того, как мы добавили эту ссылку, мы получали 5 электронных писем в день с жалобами на эту проблему. Теперь – ни одного.

Отвечайте быстро

Быстрое время ответа на запросы к техподдержке должно быть главным приоритетом

Пользователи загораются, когда вы быстро отвечаете на их вопросы. Они так привыкли к “консервированным” ответам, которые приходят несколькими днями позже, если вообще приходят – так что вы можете серьезно выделиться среди конкурентов, предлагая продуманные ответы сразу. В рабочее время мы отвечаем на 90% запросов по электронной почте в течение 90 минут - а часто и в течение получаса. И пользователи в восторге от этого.

Даже если у вас нет совершенного ответа, ответьте что-нибудь. Если кто-то жалуется на проблему, которая не может быть устранена немедленно, скажите что-то вроде: “Мы слышим, что вы говорите, и мы будем работать над этим в будущем”. Это – хороший способ развеять потенциально неприятную ситуацию.

Пользователям нравится прямота, и часто их недовольство на глазах превращаются в вежливость, когда вы отвечаете быстро и прямо.

Армия многих

Как может маленькая команда, состоящая всего из трех разработчиков, создавать новый продукт и успешно конкурировать с большими компаниями? Ответ состоит в том, чтобы быть армией многих.

С самого первого дня вашей компании помните, что ваши пользователи – это ваш самый главный ресурс, они жизненно важны для вашего долгосрочного успеха, и потому оказывайте им королевские почести. Вы можете конкурировать с большими компаниями, если вы начинаете с малого и уделяете внимание каждому из ваших пользователей.

Ваши первые пользователи – это те, кто укажет вам на на ошибки в программе и на те нужды пользователей, которым программа не удовлетворяет, и именно они расскажут о вашей программе другим.

Это не значит, что ваш программный продукт должен быть совершенным в момент запуска. Наоборот, выпускайте новые версии программы рано и часто. Однако, когда ваши пользователи сообщают об ошибках, обязательно сразу же ответьте им и поблагодарите за отзывы.

Пользователи не рассчитывают на то, что ваш продукт будет совершенным, и что он будет содержать все необходимые функции. Тем не менее, пользователи рассчитывают на то, что вы прислушиваетесь к ним и заботитесь о них, так что продемонстрируйте вашу заботу. Забота в большом дефиците у большинства крупных компаний, так что развивайте чувство сообщества как можно раньше.

В компании Blinklist мы отвечаем на каждое электронное письмо от клиента, обычно в течение часа (если мы в это время не спим). У нас также есть форум, где ни один запрос или комментарий не остается без ответа.

Одинаково важно то, что все наши разработчики получают обратную связь от пользователей и активно участвуют в дискуссиях на форуме. Таким образом, мы медленно, но верно строим активное и лояльное сообщество Blinklist.

—Майкл Рейнинг(Michael Reining), соучредитель компаний MindValley и Blinklist

Трудная любовь

Будьте готовы сказать “нет” своим пользователям

Когда дело касается запросов и пожеланий о добавлении новых функций к программе, клиент не всегда прав. Если бы мы добавили каждую функцию, когда-либо предложенную нашими пользователями, наши продукты были бы никому не нужны.

Если бы мы потакали каждому капризу наших пользователей, программа Basecamp имела бы: обширные функции учета времени, обширные функции выставления счетов, обширные функции создания расписаний, обширные функции календаря, обширные функции отслеживания зависимостей между задачами, обширные функции обмена мгновенными сообщениями, обширную вики-функциональность и обширное все-что-вы-только-можете-себе-представить.

И в то же время запрос №1 наших пользователей состоит в том, чтобы мы сохранили программу Basecamp простой.

Вот еще пример: несмотря на жалобы, мы решили не поддерживать IE5 в наших продуктах. Это составляло 7% нашего рынка. Тем не менее мы решили, что более важно позаботиться об остальных 93%. Исправление ошибок и тестирование под IE5 просто не стоило потраченного времени. Уж лучше мы будем совершенствовать наш продукт для других пользователей.

Как компания-разработчик программного обеспечения, вы должны вести себя как фильтр. Не все, что кто-либо предлагает, является правильным ответом. Мы рассматриваем все запросы, но потребитель не всегда прав. Бывают времена, когда вам просто приходится кого-то разозлить. C’est la vie.

Добавим к вышесказанному: крайне важно, чтобы вы, как компания-разработчик, любили свой продукт. А вы не сможете его любить, если он наполнен кучей вещей, с которыми вы не согласны. И это еще один довод в пользу того, чтобы отказать пользователям в выполнении запросов, в необходимость которых вы не верите.

В хорошей компании

Используйте форум или чат, чтобы пользователи имели возможность помогать друг другу

Форумы и сетевые групповые чаты – хороший способ дать пользователям возможность задавать вопросы и помогать друг другу. Путем убирания посредника – то бишь вас – вы обеспечиваете открытый поток общения и экономите свое время.

На форумах наших продуктов пользователи предлагают идеи и решения, оставляют запросы, истории и многое другое. Мы появляемся время от времени, чтобы предложить помощь – но прежде всего, форумы – это место, где люди помогают друг другу и делятся своим опытом работы с продуктом.

Вы будете удивлены тому, насколько люди хотят помогать друг другу.

Публикуйте ваши неудачи

Выпустите плохие новости и уберите их с дороги

Если что-либо пошло не так, расскажите людям. Даже если они не заметили.

Например, сайт Basecamp однажды не работал в течение нескольких часов среди ночи. 99% наших пользователей никогда бы об этом и не узнали, но мы все равно поместили сообщение о непредвиденном сбое в нашем блоге Everything Basecamp. Мы считаем, что наши пользователи заслуживают того, чтобы об этом знать.

Вот пример нашего сообщения в случае неполадок: “Мы приносим свои извинения за сбои в работе сайта сегодня утром. У нас были проблемы с базой данных, что привело к замедлению работы программы и ее сбоям для некоторых пользователей. Мы решили эту проблему и работаем над тем, чтобы она больше не повторялась. Благодарим за терпение и еще раз просим прощения за неполадки.”

Будьте открыты, честны и прозрачны, как только возможно. Не держите секретов и не прячьтесь. Информированный пользователь – ваш лучший пользователь. Кроме того, вы поймете, что ваши неудачи в глазах ваших пользователей не так уж и плохи. Пользователи обычно счастливы предоставить вам пространство для маневра, когда знают, что вы с ними честны.

Заметим еще по поводу новостей, плохих и хороших. Когда появляются плохие новости, выпустите их сразу. Хорошие новости, с другой стороны, выпускайте медленно. Если вы можете продлить прекрасные чувства, сделайте это.

Будьте оперативны, прямы и честны

Это может показаться странным, но лучший вариант развития событий – когда компания сама сообщает плохие новости. Такой проактивный подход избавит вашу компанию от попадания в ослабленное, оборонительное положение.

—Грег Шервин(Greg Sherwin), Вице-президент по технологиям приложений (Vice President of Application Technology) компании CNET, и Эмили Авила(Emily Avila), руководитель компании Calypso Communications (из книги A Primer for Crisis PR)

После выпуска глава 15

Настройка через месяц

Выпустите обновление через 30 дней после выпуска

Быстрое обновление показывает движение. Показывает, что вы прислушиваетесь к советам пользователей. Показывает, что у вас еще есть "порох в пороховницах". Оно дает вторую волну разговорам. Оно подкрепляет первоначальные положительные эмоции, связанные с вашим продуктом. Оно дает пищу для обсуждения и сообщений в блогах.

Мысли о грядущем быстром обновлении также помогают вам перед выпуском сосредоточиться на наиболее критических компонентах. Вместо того, чтобы втискивать в программу новые и новые детали, вы можете просто совершенствовать имеющиеся. Потом вы сможете выпустить продукт “в люди”. А когда он уже там, вы начнете собирать отзывы пользователей и узнаете, какие области потребуют внимания на следующем этапе.

Этот подход с маленькими шажками хорошо оправдал себя в случае Backpack. Вначале мы выпустили базовый продукт, а потом, несколькими неделями позже, добавили новые функции, такие как Backpack Mobile для карманных компьютеров и поддержку тэгов, так как именно эти функции запрашивались клиентами чаще всего.

Продолжайте выпуск сообщений

Покажите, что ваш продукт живет – продолжайте блог продукта после выпуска

Не переставайте писать в блог после выпуска продукта. Покажите, что ваш продукт живет, с помощью блога, который вы часто обновляете (как минимум раз в неделю, а если можете – то и чаще).

Что туда можно включить:

- Частые вопросы
- Как работать с программой
- Советы и решения
- Новые функции, обновления, исправление ошибок
- Разговоры/пресса

Блог не только показывает, что ваша программа живет, он еще делает вашу компанию более человечной. Еще раз, не бойтесь держать тон дружественным и личным. Маленькие компании иногда считают, что им надо все время выглядеть большими и сверх-профессиональными. Это похоже на бизнес-версию комплекса Наполеона. Не бойтесь выглядеть небольшими. Радуйтесь тому, что можете говорить с клиентами как с друзьями.

Он живет

Часто обновляемый блог продукта – лучшее свидетельство тому, что продукт находится в активной разработке, его любят, и свет в его окошке не гаснет. Заброшенный блог – свидетельство заброшенного продукта, знак того, что его водители уснули за рулем.

Поддерживайте общение с пользователями в блоге продукта, будьте прозрачны и щедры в подаче информации. Пусть философия вашей компании будет видна. Выложите ссылки на конкурентов и открыто обсуждайте их. Анонсируйте добавление новых функций и держите комментарии открытыми для обратной связи.

Продукт живет, когда он говорит со своими пользователями и слушает их. Часто обновляемый блог продукта поддерживает прозрачность, чувство сообщества и лояльности к вашей марке. Дополнительно вы получаете бесплатную рекламу.

Как редактор Lifehacker, я постоянно просматриваю блоги любимых приложений, таких как Google, Flickr, Yahoo, del.icio.us и 37signals. И я большей вероятностью буду упоминать их, чем те, которые высылают односторонние пресс-релизы из ниоткуда и не поддерживают общения со своими пользователями и поклонниками.

—Джина Трапани (Gina Trapani), веб-разработчик и редактор Lifehacker, путеводитель по производительности и программному обеспечению

Не бета, а лучше

Не используйте “бета” в качестве козла отпущения

В наши дни кажется, что все постоянно находится в бета-версии. Неумирающая бета-версия говорит пользователям, что вы не так уж и хотите выпускать завершенный продукт. Она говорит: “Пользуйтесь вот этим, но если оно несовершенно – это не наша вина”.

Бета сваливает все на пользователя. Если вы сами не уверены в вашей версии, то как клиенты могут быть в ней уверены? Приватные бета-версии – это нормально, а общедоступные – это фигня. Если нечто недостаточно хорошо для всеобщего потребления – не давайте это публике.

Не ждите, что ваш продукт достигнет совершенства. Этого не случится. Возьмите на себя ответственность за то, что вы выпускаете. Выпустите это и назовите новой версией. В противном случае вы просто ищете отговорки.

Бета ничего не значит

Можете обвинять Google со товарищи в таких проблемах. Сейчас пользователи научены программистским сообществом, что понятие “бета” в действительности ничего не значит.

—Мэри Ходдер (Mary Hodder), information architect and interaction designer (from The Definition of Beta)

Все время

Это только я такой, или мы все находимся в бета-версии, все время?

—Джим Кудал (Jim Coudal), основатель компании Coudal Partners

---

Не все ошибки в программе созданы равными

Разделите ваши ошибки по приоритетам (и даже проигнорируйте некоторые из них)

Если вы нашли ошибку в вашем продукте – это не повод впадать в панику. Все программы содержат ошибки, это просто неоспоримый факт.

Не нужно сразу же исправлять каждую ошибку. Большинство ошибок надоедливы, но не смертельны. Те, которые надоедливы, могут быть отложены. Ошибки типа “что-то тут выглядит не так” и подобные можно без ущерба отложить на некоторое время. А уж если ошибка ломает вашу базу данных – тогда, конечно, она должна быть исправлена немедленно.

Определите приоритет каждой ошибки. Сколько пользователей от нее страдают? Насколько серьезна проблема? Заслуживает ли ошибка немедленного внимания, или может подождать? Что можно сделать прямо сейчас, чтобы помочь наибольшему количеству пользователей? Часто добавление новой функции может быть более важным, чем исправление существующей ошибки.

Также не создавайте ореол страха вокруг ошибок. Они случаются. Не ищите постоянно, кого бы обвинить. Меньше всего вам нужно, чтобы ошибки прятались под ковер, вместо того, чтобы открыто обсуждаться.

И помните то, что мы говорили ранее о важности честности. Если клиенты жалуются на ошибку, будьте с ними честны. Скажите, что вы заметили эту проблему и работаете над ней. Если вы не собираетесь работать на ней прямо сейчас, объясните, что вы заняты теми областями продукта, которые помогут большему числу пользователей. Честность – лучшая политика.

Переждите шторм

Подождите, пока страсти улягутся, перед тем как действовать

Если раскачивать лодку, появляются волны. Когда вы добавляете новую функцию, изменяете правила или удаляете что-нибудь – реакции будут разными, и часто отрицательными.

Избегайте паники и желания быстро все поменять в ответ. Да, вначале страсти кипят. Но если вы переждете этот начальный период, который составляет 24-48 часов, все уляжется. Большинство пользователей реагируют на изменения до того, как они действительно изучили и использовали то, что вы добавили (или смирились с тем, что вы удалили). Так что расслабьтесь, и не двигайтесь, пока не пройдет некоторое время. И тогда уж вы сможете предложить более продуманный ответ.

Помните также, что отрицательные реакции почти всегда звучат громче, чем положительные. Вам даже может показаться, что вы слышите только отрицательные отзывы, тогда как большинство пользователей довольно изменениями. Убедитесь, что вы не отказываетесь глупо от необходимого, но противоречивого решения.

Не отставайте от соседей

Подпишитесь на новости о ваших конкурентах

Подпишитесь на новости и о своих продуктах, и о продуктах конкурентов (это всегда мудро – следить за передвижениями противника). Используйте сервисы, такие как PubSub, Technorati, Feedster и другие, чтобы быть в курсе (в качестве ключевых слов используйте названия компаний и продуктов). С помощью RSS эта постоянно меняющаяся информация будет доставлена прямо к вам, так что вы будете всегда в курсе.

Остерегайтесь монстра разбухания

Более зрелый - не значит более сложный

С развитием событий не бойтесь противостоять разбуханию. Всегда будет соблазн расширять продукт в объеме. Но это делать не обязательно. То, что продукт растет и становится более зрелым – не должно значить, что и более сложным.

Не обязательно быть ручкой для письма в открытом космосе, которая пишет вверх ногами. Иногда можно просто быть карандашом. Не нужно становиться швейцарским армейским ножом. Можно быть просто отверткой. Не нужно создавать часы для ныряния на 5000 метров, если ваши потребители – люди сухопутья, которые просто хотят узнать, который час.

Не раздувайте просто ради раздувания. Именно так получаются разбухшие программы.

“Новое” не обязательно значит “улучшенное”. Иногда наступает точка, когда лучше просто оставить продукт как есть.

Это – одно из главных преимуществ построения сетевого программного обеспечения по сравнению с традиционным. Производители традиционного программного обеспечения, такие как Adobe, Intuit и Microsoft, должны каждый год продавать вам новые версии. И, так как они не могут просто продать вам новую версию, они должны оправдать расходы добавлением новых функций. Здесь и начинается разбухание.

В случае сетевого программного обеспечения, основанного на модели подписки, клиенты платят помесячно за право использования сервиса. Вам не нужно продавать им новые версии путем C4;обавления новых и новых свойств, нужно только обеспечивать текущий сервис, значимый для них.

Двигайтесь по течению

Будьте открыты для новых путей и изменения направлений

То, что придает красоту сетевым приложениям – это их изменяемость. Вы не упаковываете программу в коробочку, поставляете пользователю, а потом ждете новой версии в течение многих лет. Напротив, вы можете вносить изменения по пути. Примите мысль, что ваша первоначальная идея могла быть не самой лучшей.

Посмотрите на проект Flikr. Он начинался как онлайновая игра для нескольких игроков, которая называлась “Бесконечная Игра” (The Game Neverending). Создатели проекта быстро поняли, что возможность совместного использования фотографий была куда более значимой, чем сама игра (которая в конце концов оказалась на полке). Учитесь признавать свои ошибки и менять курс.

Будьте серфером. Наблюдайте за океаном. Смотрите, где образуются большие волны, и соответственно корректируйте курс.

Заключение глава 16

Заводите моторы

Готово!

Ну хорошо, вы это сделали! Надеемся, вы с нетерпением ждете момента, чтобы начать применять Getting Real в своих программах. Сейчас самое лучшее время, чтобы создавать отличные программы, используя минимум ресурсов. При наличии хорошей идеи, страсти, времени и навыков – выше только небо.

Несколько мыслей в заключение:

Воплощение

Каждый может прочесть книгу. Каждый может придумать идею. У каждого есть родственник, работающий веб-дизайнером. Каждый может завести блог. Каждый может нанять кого-то еще, чтобы вместе наваять какой-то код.

Разница между вами и всеми остальными в том, как хорошо вы воплощаете. Успех достигается именно безупречным воплощением.

При создании программного обеспечения это значит, что нужно правильно воплотить множество вещей. Вы не можете просто написать хорошую спецификацию, а потом не выполнить своих обещаний. Безупречный дизайн интерфейса не поможет, если в коде полно трюков. Самая замечательная программа многого не стоит, если в результате плохого продвижения о ней никто так и не узнал. Чтобы преуспеть, нужно скомбинировать все эти элементы.

Главное – сохранять равновесие. Если вы гнетесь слишком далеко в каком-то направлении – вы на пути к падению. Постоянно ищите свои слабые звенья и сосредоточьтесь на них, пока не приведете их в порядок.

Люди

Стоит еще раз повторить, что главный элемент построения успешного сетевого приложения – это люди, работающие над ним. Мантры, проектирование от эпицентра, меньшее количество кода и все остальные замечательные идеи ничего не значат, если у вас нет людей, способных воплотить эти идеи.

Вам нужны люди, страстно любящие свое дело. Люди, заботящиеся о своем искусстве – и считающие свое дело искусством. Люди, чувствующие гордость за свою работу вне зависимости от наличия денежного вознаграждения. Люди, совершенствующие продукт до самых мелких деталей, даже если 95% пользователей их не заметит. Люди, которые стремятся создать отличный продукт и на меньшее не согласны. Люди, которым нужны люди... то есть не совсем, но мы не могли не вспомнить песню Барбры Стрейзанд. В любом случае, когда вы нашли таких людей – держитесь за них. В конце концов, от людей в вашей команде зависит успех или неуспех вашего проекта – и вашей компании.

Больше, чем просто программы

Также стоить заметить, что подход Getting Real применим не только к построению сетевых программ. Когда вы ближе познакомитесь с его идеями, вы обнаружите их повсюду вокруг себя. Например:

- Силы особого назначения, такие как Зеленые Береты или Морские Котики, используют малые бригады и быстрое развертывание для выполнения задач, для которых другие подразделения слишком велики или слишком медлительны.
- Рок-группа White Stripes ограничивается в своем творчестве простыми средствами: два человека, простые песни, по-детски незамысловатая партия ударных, использование студии по минимуму и т.д.
- iPod фирмы Apple отличается от конкурирующих продуктов тем, что в него не встроены ни радио, ни диктофон.
- В американском футболе в результате быстрой атаки можно захватить большой участок поля за счет уменьшения “бюрократии” и толкотни.
- Успешные кулинарные книги и телевизионная передача Рэйчел Рэй основаны на идее 30-минутных блюд в стиле Get Real.
- Эрнест Хэмингуэй и Рэймонд Карвер использовали простой и ясный язык, который, тем не менее, производит сильное впечатление.
- Шекспир с наслаждением творил в тесных рамках сонета, четырнадцатистрочного лирического стихотворения, написанного пятистопным ямбом.
- И так далее...

Безусловно, подход Getting Real нацелен прежде всего на то, чтобы создавать хорошие программы. Но нет причин, чтобы на этом он и ограничивался. Возьмите эти идеи и попытайтесь применить их к другим аспектам вашей жизни. Может быть, вы будете приятно удивлены некоторыми результатами.

Будьте на связи

Сообщите нам, насколько полезным оказался для вас подход Getting Real. Отправьте нам письмо по электронному адресу gettingreal [at] 37signals [dot] com.

Также следите за новинками от 37signals – заходите на сайт Signal vs. Noise - наш блог о Getting Real, удобстве использования программного обеспечения, проектировании и еще о многом другом.

Спасибо за то, что прочитали нашу книгу, удачи вам!