РУКОВОДСТВО ДЛЯ НАСТОЯЩИХ
САМУРАЕВ

Гибкое управление IT-проектами

Как мастера Agile делают выдающееся ПО

Pragmatic Bookshelf

ПИТЕР

Джонатан Расмуссон

Джонатан Расмуссон

Гибкое управление IT-проектами. Руководство для настоящих самураев

Текст предоставлен правообладателем
http://www.litres.ru/pages/biblio_book/?art=8342877
Гибкое управление IT-проектами. Руководство для настоящих самураев. | Расмуссон Дж.: Питер; Санкт-Петербург; 2012
ISBN 978-5-459-01205-7

Аннотация

Прочитав эту книгу, вы получите знания и навыки, необходимые для того, чтобы запустить, проработать и успешно завершить гибкий проект, причем приведенный материал вас изрядно развеселит. Если вы – руководитель проектов, это издание станет для вас инструментом, который поможет реализовать гибкий проект от начала и до конца. Если же вы – аналитик, программист, тестировщик, разработчик пользовательских взаимодействий или проект-менеджер, книга даст вам идеи и базовые знания, необходимые для того, чтобы стать ценным членом команды разработчиков.

«Руководство для настоящих самураев» обходится без лишней теории, из-за которой другие книги совсем не отвечают духу гибкости. Она полна испытанных методов,

невыдуманных историй, приятного юмора и прикладных упражнений-руководств, которые помогут вам делать правильные вещи наилучшим способом.

Содержание

Джонатан Расмуссон Гибкое управление IT-проектами. Руководство для настоящих самураев

Как мастера Agile делают выдающееся ПО

Jonathon
The Agile Samurai Rasmusson
How Agile Masrets Deliver Great Software

The progmatic Bookshelf

несет ответственности за возможные ошибки, связанные с использованием книги.

Об авторе

Джонатан Расмуссон, опытный предприниматель и бывший консультант по гибкой разработке (agile coach) в компании ThoughtWorks, занимается консалтингом на международном уровне, помогая клиентам находить наилучшие пути совместной работы и взаимодействий. Если Джонатан не едет на мотоцикле на работу по лютому канадскому морозу, то в свободное время он ведет блог http://agilewarrior.wordpress.com, где делится своим опытом из области гибкой разработки.

Благодарности

Эта книга никогда бы не получилась, если бы не поддержка Таннис – любви всей моей жизни – и трех наших чудесных детей, Лукаса, Роуэна и Бринна, которые прошли вместе со мной весь этот путь.

Подобная книга также не могла бы появиться без участия чудесного редактора и издателя. Сюзанна Пфальцер достойна всяческих похвал.

Следует также упомянуть тех разработчиков-первопроходцев, на труд которых я опирался при написании книги. Это Кент Бек, Мартин Фаулер, Рон Джеффрис, Боб Мартин, Джошуа Кериевски, Том и Мэри Поппендик, Кэти Сьерра и чудесный коллектив компании ThoughtWorks.

И, разумеется, эта книга никогда бы не состоялась, если бы не невероятно плодотворное сотрудничество и вдумчивый подход, проявленные рецензентами и комментаторами: Ноэлем Раппином, Аланом Френсисом, Кевином Гиси, Джессикой Уотсон, Томасом Гендроном, Дэйвом Клейном, Майклом Сикорским, Дэном Нортом, Джанет Грегори, Санджай Манчиганти, Венди Линдеманн, Джеймсом Эвери, Робином Даймондом, Томом Поппендиком, Элис Тот, Иэном Дизом, Меган Армстронг, Рамом Сваминатаном, Хэзер Карп, Чедом Фурнье, Мэттом Хьюзом, Майклом Менардом, Тони Семана, Ким Шриер и Рихейлом Кристофом.

Особая благодарность Ким Уимпсетт и Стиву Питеру за превосходное техническое редактирование и набор текста.

Спасибо вам, мама и папа, за то, что любите и вдохновляете меня.

Также спасибо Дэйву и Энди за их прекрасную компанию, в которой молодые и целеустремленные авторы могут творить и делиться своей работой со всем миром.

Самурай гибкой разработки – это неистовый программист-профессионал, умеющий справляться с самыми сложными софтверными проектами и укладываться в практически нереальные сроки, делая это легко и изящно.

Мастер-сэнсэй

Рад встрече с вами

Гибкая методология разработки программ (agile) напоминает нам, что, хотя компьютеры и исполняют код, его написанием и поддержкой занимаются все-таки люди.

Гибкая разработка – это концептуальный каркас, мировоззрение и подход к созданию программ. Она отличается экономностью, быстротой и прагматизмом. Это, конечно, не палочка-выручалочка, но все же такая парадигма радикально повышает шансы на успех, стимулируя вашу команду выдавать максимум, на который она способна.

В книге описывается реализация проекта, в котором применяется методология гибкой разработки. Хочу сказать, что такой проект действительно должен превосходить все ожидания. Ваши проекты будут заканчиваться вовремя и полностью укладываться в бюджет. Кроме того, клиенты будут полностью довольны создаваемыми для них программами и с удовольствием дадут вам новые заказы и примут посильное участие в работе.

Внутри коллектива вам предстоит освоить определенные навыки.

♦ Умение успешно подготавливать и запускать с пол-оборота собственный гибкий проект. Процесс должен быть настолько ясен, чтобы у вас вообще не возникало вопросов из области «на какой стадии находится работа?» и «как это на-

зывается?».

♦ Умение собирать требования, оценивать и планировать работу прозрачно, открыто и честно.

♦ Умение работать с жаром. Вы узнаете, как превратить свой гибкий проект в отлаженный механизм, бесперебойно производящий высококачественный код, готовый к реальному использованию.

Если вы руководитель проекта, эта книга послужит инструментом для подготовки собственного гибкого проекта и ведения его от начала до конца. Если вы аналитик, программист, тестировщик, разработчик взаимодействия с пользователем, то сможете почерпнуть из книги идеи и базовые знания, необходимые для того, чтобы стать полноправным членом команды, занимающейся гибкой разработкой.

Как читать эту книгу

Можете свободно переходить к чтению любой главы, которая вас заинтересует. Но если вы хотите правильно построить процесс работы с самого начала, рекомендую прочесть книгу от начала и до конца.

В части I дается краткий обзор гибкой методологии разработки и объясняется, как работают команды, использующие данную парадигму.

В части II рассказывается об одной из наиболее многообещающих деталей предлагаемого метода – о подборе средств, которые составят арсенал вашей команды при реализации проекта. Речь пойдет о так называемой стартовой колоде (inception deck).

Часть III посвящена отзывам пользователей о проектах, выполненных в режиме гибкой разработки. Здесь же рассказано, как построить первый свой план такого проекта.

В части IV речь идет о выполнении работы. Именно отсюда вы узнаете, как из плана получается нечто реальное – готовая программа, с которой может работать клиент.

Наконец, в части V подытоживается все сказанное. Здесь описаны основные гибкие методы написания ПО, которым нужно следовать, чтобы постоянно повышать качество своих программ и снижать расходы на их текущую поддержку.

Неслучайный юмор

Не относитесь к этой книге с излишней серьезностью – желательно, чтобы вы изучали изложенный материал с чувством юмора.

Для этого текст сопровождается иллюстрациями, отступлениями и даже анекдотами, которые помогают понять, каково это – заниматься гибкой разработкой.

Боевые истории – это «фронтовые сводки» о работе над реальными гибкими проектами, а также впечатления о некоторых успехах (и неудачах), которые пришлось пережить мне и моим «братьям по оружию» в этом драматичном деле – гибкой разработке.

Упражнения под заголовком «*А теперь попробуем*» приглашают вас оторваться от чтения, немного поразмышлять и попробовать себя на практике.

А вот и *Мастер-сэнсэй* – легендарный гуру гибкой разработки, мудрый и опытный во всех аспектах рассматриваемой парадигмы.

Мастер-сэнсэй и воин, постигающий искусство

Он будет вашим проводником и духовным наставником в странствии по гибкой разработке, время от времени обращая ваше внимание на важные принципы работы, например:

Принцип гибкой разработки

На создание готовой программы должно уходить минимум времени – от пары недель до пары месяцев. И предпочтительнее, чтобы это был максимально короткий срок.

Принцип гибкой разработки

На создание готовой программы должно уходить минимум времени — от пары недель до пары месяцев. И предпочтительнее, чтобы это был максимально короткий срок.

Мастер-сэнсэй будет делиться с вами своими глубокими озарениями и напутствиями, как применять на практике методы гибкой разработки.

Ресурсы в Интернете

У англоязычного издания этой книги есть собственный сайт http://pragprog.com/titles/jtrap, где подробнее рассказано о книге. Работать с ним можно следующим образом:

♦ участвовать в дискуссиях на форумах, обмениваться мнениями с другими читателями, энтузиастами гибкой разработки и со мной;

♦ помогать совершенствовать книгу. Для этого нужно сообщать о найденных ошибках, в том числе о фактических недочетах в материале и об опечатках.

Итак, начнем.

От издательства

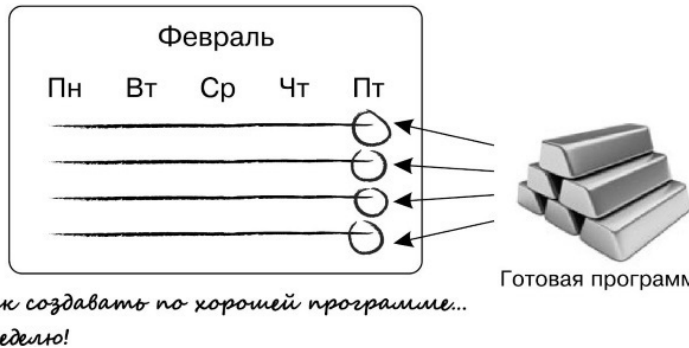
Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты vinitski@minsk.piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

Часть I
Введение в гибкую разработку

Глава 1
Сущность гибкой разработки



Что нужно, чтобы каждую неделю создавать еще одну полезную программу?

На этот вопрос я попытаюсь ответить в данной главе. Вы узнаете, как выглядит процесс написания программы с точки зрения клиента. Также будет показано, как много ненужных вещей мы зачастую преподносим заказчику и как ча-

сто нам не удается сделать то, что действительно важно, – регулярно предоставлять клиенту хорошие функциональные программы.

К концу главы вы в общих чертах поймете, как планируется гибкая разработка, как судить об успехе проекта и как всего три простые истины помогают с честью выдерживать самые сжатые сроки и реализовывать сверхсложные проекты.

1.1. Как создавать что-то полезное каждую неделю

Давайте на секунду отвлечемся от гибкой разработки и поставим себя на место клиента. Это ваши деньги и ваш проект, для реализации которого вы наняли команду профессионалов высшего класса.

Что поможет вам убедиться в том, что приглашенная команда действительно не сидела сложа руки? Кипа документов, планов и отчетов? Или регулярное еженедельное предоставление полнофункциональных, протестированных программ, включающих только самое нужное?

Итак, если вы уже начали смотреть на программу с точки зрения клиента, значит, наши дела пойдут хорошо.

1. Нужно разбить крупную проблему на мелкие задачи.

Неделя – это ведь совсем немного. Казалось бы, нельзя успеть выполнить за неделю какую-либо работу. Чтобы дело пошло, нужно разбить большую и неподъемную с виду задачу на маленькие, более простые и управляемые.

2. Нужно сосредоточиться только на том, что действительно важно, и забыть обо всем остальном.

Большая часть того, чем мы занимаемся при разработке программ, не представляет никакой или почти никакой пользы для нашего клиента.

Разумеется, нам понадобится документация. Конечно, не обойтись и без планов. Но они нужны только как вспомогательные средства на пути к созданию готовой программы.

Создавая что-то стоящее каждую неделю, вы просто вынуждены собраться и отбросить все, что не приносит пользы. В результате вы как будто отправляетесь в дорогу налегке, захватив с собой только самое необходимое.

3. Нужно убедиться, что создаваемый вами продукт ра-

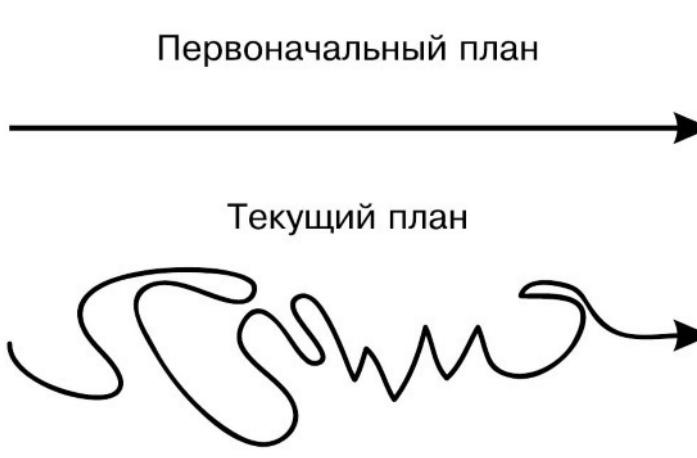
ботает.

Создание чего-то полезного каждую неделю подразумевает, что плоды вашего труда должны быть качественными. Для этого необходимо тестирование – как можно больше и как можно раньше. Прошли те времена, когда все лишнее из проекта убирали только в его финале. Теперь ежедневное тестирование уже становится образом жизни. Вся ответственность лежит на вас.

4. Работа требует участия клиента.

Чтобы достичь цели, нужно регулярно останавливаться и сверять курс с клиентом. Его участие можно сравнить со светом фар, который рассекает густой туман, когда вы несетесь по трассе со скоростью 100 км/ч. Без таких консультаций клиент не может отслеживать вашу работу – в результате вы оба можете попасть в кювет.

5. При необходимости нужно изменять курс.



При работе случается всякое. Положение изменяется. То, что неделю назад казалось важным, сегодня может быть отбраковано. Если выстроить план и слепо ему следовать, вы не сможете справиться с непредвиденными обстоятельствами в случае их возникновения. Поэтому, если реальность нарушает ваши планы, меняйте планы, а не реальность.

6. Вы берете на себя ответственность.

Когда вы ставите перед собой цель каждую неделю создавать что-то стоящее и отчитываться перед клиентом, на что тратите его деньги, вы берете на себя определенную ответственность.

- ♦ Вы отвечаете за качество.
- ♦ Вы придерживаетесь расписания.

♦ Вы устанавливаете определенную планку.
♦ Вы тратите средства так, как если бы они были вашими.

Считаю ли я, что настанет день, когда все будут работать именно так? Ни в коем случае – ведь я же не удивляюсь тому, что большинство людей питается неправильно и не утруждает себя физкультурой.

Создание чего-то ценного каждую неделю – дело не для слабонервных. Выбирая такой подход к работе, вы словно попадаете в луч прожектора. В нем никуда не спрятаться. Ва-

ши творения либо полезны, либо нет.

Но если вам нравится быть на виду, вы сторонник качественной работы и вас распирает желание действовать, то именно для вас работа в команде специалистов, применяющих гибкую методологию, может быть не только очень плодотворной, но и чертовски интересной.

А если вас все же пугает перспектива работать в темпе «на все про все неделя» – не отчаивайтесь. Большинство команд, работающих в гибком режиме, начинает с двухнедельных проектов (а если команда очень большая – то с трехнедельных).

Это просто метафора, суть которой сводится к тому, что вы должны регулярно предоставлять клиенту готовые программы, и для этого требуется определенная отдача с его стороны, а при необходимости – изменение курса. Вот и все.

Принцип гибкой разработки

Наш наивысший приоритет — заслужить доверие клиента, регулярно и последовательно предоставляя ему хорошие программы.

1.2. Как происходит планирование при гибкой разработке

Планирование проекта в гибкой манере схоже с подготов-

кой к долгим насыщенным выходным. Не буду писать длинных списков необходимых дел и задач, лучше давайте поговорим о таких нетривиальных вещах, как *журнал пожеланий к продукту и пользовательские истории*.

В гибкой разработке под журналом пожеланий (master story list)[1] понимается список задач, которые предстоит решить программисту. В нем упоминаются все важнейшие функции (пользовательские истории, user stories) – пожелания, предъявляемые клиентом к программе. Приоритетность тех или иных функций определяет сам пользователь, ваша команда разработчиков оценивает эти приоритеты и

[1] В английских источниках также употребляются термины backlog и product backlog, в русском языке встречается понятие «бэклог». – *Примеч. пер.*

закладывает базовый план проекта.

Механизм выполнения всех задач в гибком проекте – это так называемая *итерация*. Под итерацией понимается короткий одно– или двухнедельный период, в ходе которого команда берет важнейшие пользовательские истории и преобразует их в действующие, протестированные программные функции.

Члены вашей команды могут прикинуть, сколько работы способен взять на себя каждый из них, измеряя *скорость работы команды* (сколько дел можно выполнить за одну итерацию). Отслеживая скорость работы команды и используя эти данные для того, чтобы спрогнозировать, сколько удастся сделать в будущем, вы сможете ставить реальные сроки и соблюдать их, а ваша команда не будет брать на себя чрезмерных обязательств.

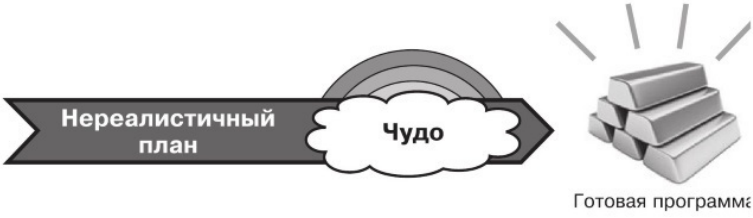
Когда оказывается, что вам и вашему клиенту предстоит сделать слишком много, ваш единственный выход – сделать меньше. *Гибко подходя к объему задач*, вы сможете сохранить сбалансированные планы, а ваши обещания останутся реалистичными.

И если реальность идет вразрез с планом, нужно менять план. *Адаптивное планирование* – это краеугольный камень гибкой разработки.

Пока это все, что требуется сказать о гибком планировании. Такой метод планирования будет подробнее рассмотрен в главе 8.

Если сроки поджимают, то, как говорится, «надо – значит надо». Просто советую убедиться, что вы жертвуете собой ради стоящего дела, а не ради совершенно нереального обещания, данного примерно год назад при обзоре показателей эффективности работы.

Действительно, нереальные обещания время от времени даются и команды разработчиков сталкиваются с необходимостью сотворить невозможное. Но это неправильно. Работать с расчетом на чудо – не лучший способ реализации проекта и еще более порочный принцип по отношению к ожиданиям клиента.



При гибкой разработке необходимость в чудесах такого рода отпадает, так как вы собираетесь работать с клиентами открыто и честно с самого начала – рассказываете, что предстоит сделать, и позволяете принимать осознанные решения о функционале программы, финансировании и сроках.

Все дело в выборе. Можете продолжать веровать в миф о том, что все раз – и получится. Или можно разработать вместе с клиентом такой план, в который поверите и вы и он.

1.3. Сделано – значит сделано

Допустим, ваши дедушка с бабушкой за небольшое вознаграждение попросили соседского сына-подростка сгрести граблями опавшие листья во дворе на даче, сложить в мешок и отнести в лес. Сочтут ли дедушка с бабушкой работу выполненной, если парень сделает что-то из следующего:

- ♦ составит отчет о том, как он спланировал работу с граблями;
- ♦ предложит элегантный метод работы;

♦ составит тщательный и полный план тестирования?

Ничего подобного! Парень не получит ни копейки, пока не уберет листья, не уложит их в мешок и не отнесет куда следует.

При гибкой разработке применяется тот же принцип. В данном случае под реализацией функции понимается решение всех задач, необходимых для получения готового к работе кода.

Анализ, проектирование, написание кода, тестирование, разработка и дизайн уровня взаимодействия с пользователем (и проверка удобства этого уровня) – здесь перечислено все. Это не означает, что уже в первой версии программы у нас будут готовы все прибамбасы или что в конце каждой итерации у нас будет получаться готовый продукт. Но мы собираемся к этому стремиться.

Если с продуктом нельзя работать на практике – это озна-

чает, что он не готов. И именно поэтому разработчик, исповедующий гибкую методологию, должен серьезно к ней относиться и понимать три простые истины.

Принцип гибкой разработки

Основной критерий успеха — практическая применимость программы.

1.4. Три простые истины

Далее описаны три простые истины, действующие при руководстве проектами. Если руководствоваться ими, удается справиться со значительной долей нервозности и сбоев, с которыми регулярно приходится сталкиваться в софтверных проектах.

Три простые истины

1. Невозможно собрать все необходимые требования еще до начала проекта.

2. Любые требования, которые удастся собрать, обязательно изменятся.

3. На реализацию проекта обязательно понадобится больше времени и денег, чем было запланировано.

Принимая первую истину, мы перестаем бояться предстоящего проекта, который еще не известен нам во всех деталях. Мы понимаем, что требования еще предстоит разузнать и что работу вполне можно начинать еще до окончания сбора всей информации.

Осознавая вторую истину, мы перестаем бояться перемен или избегать их. Мы знаем, что рано или поздно они наступят. Принимаем их такими, какие они есть. И при необходимости корректируем наши планы в соответствии с ними.

Наконец, соглашаясь с третьей истиной, мы уже не драматизируем, если не успеваем сделать все запланированное в установленные сроки и испытываем недостаток ресурсов. Это нормальное состояние для любого интересного проекта. Вы делаете единственную вещь, которую можете сделать, – устанавливаете приоритеты, выполняете самую важную работу в первую очередь, а менее важную оставляете на потом.

Усвоив три эти простые истины управления проектами, вы увидите, что львиная доля стресса и волнений, традиционно связанных с написанием программ, куда-то исчезает. Вы сможете думать и изобретать с такой степенью сосредоточенности и ясности, какой не может похвастаться практически никакая другая область промышленности.

И не забывайте…

Путей всегда много!

Как и мороженое, гибкая разработка может иметь разные оттенки вкуса.

♦ У вас есть скрам (Scrum) – этот метод управления охватывает гибкий проект как оболочка.

♦ У вас есть экстремальное программирование (Extreme Programming, XP) – требующие высокой дисциплины основные практики разработки программ, жизненно важные для любого гибкого проекта.

♦ У вас есть бережливая разработка (Lean) – сверхэффективный аналог производственной системы, нацеленный на постоянную оптимизацию рабочего процесса (такая философия принята в компании «Тойота»).

А потом у вас появляется собственный гибкий метод. Например, такой метод может пригодиться для поиска выхода из ситуации, когда вы проехали со своей семьей на машине пару тысяч километров и обнаружили, что парк развлечений, в который вы направлялись, закрыт на ремонт.

Эта книга, как и вся остальная литература по гибкой разработке, – обычный обмен опытом. В ней рассказано о методах, которые я и другие авторы подобных книг нашли полезными при работе с клиентами. В данном издании я буду делиться с вами находками и инновациями, относящимися ко всем направлениям гибкой разработки, а несколько подобных методов нам придется изобрести самостоятельно. Читайте, изучайте, ставьте перед собой задачи и берите от книги именно то, что вам требуется.

Но учитывайте, что ни одна книга или метод не дадут вам ответов на все вопросы – в любом случае что-то придется додумывать самостоятельно. Каждый проект особенный, и хотя определенные принципы и практики будут действовать всегда[2], именно от вашей уникальной ситуации и контекста работы будут зависеть тонкости их применения.

Несколько слов о терминологии

В большинстве гибких методологий термины уже в основном устоялись, но некоторые понятия по-разному называются в скраме и экстремальном программировании.

В книге я постараюсь соблюдать единство терминологии (вообще, мне ближе экстремальное программирование), но хочу оговориться, что следующие понятия равнозначны и являются синонимами:

- *итерация* (iteration) – то же, что и *спринт* (sprint);

[2] http://agilemanifesto.org.

♦ *журнал пожеланий* (master story list) – то же, что и *бэклог* (product back log);

♦ *клиент* (customer) – то же, что и *владелец* (product owner)[3].

Что дальше?

Итак, мы познакомились с основами. Теперь пришло время включить вторую передачу и поговорить о том, что такое команда.

В следующей главе, которая рассказывает о командах гибких разработчиков, мы рассмотрим, как должна быть построена такая команда, как она должна работать над проектом, а также о некоторых вещах, которые каждый член команды должен уяснить до начала проекта.

[3] «Владелец» – термин условный. Обычно так называют заказчика или его представителя, который определяет требования к продукту. В agile-команде эту роль может исполнять менеджер проекта, бизнес-аналитик или клиент. – *Примеч. ред.*

Глава 2
Знакомство с командой разработчиков

Команда разработчиков при гибком методе работы – это нечто совершенно особенное. В типичном гибком проекте нет жестко заданных ролей. Все могут делать что угодно. И

все же при всем хаосе, кажущейся неразберихе и отсутствии формальной иерархии отлаженным командам гибких разработчиков как-то удается регулярно создавать классные программы.

В этой главе будет подробно рассмотрено, что обеспечивает работу гибкой команды. Мы изучим характеристики хороших команд, построенных по такому принципу, различия между гибкими командами, а также обсудим несколько рекомендаций, упрощающих поиск квалифицированных сотрудников.

К концу главы вы будете представлять, как выглядит типичная гибкая команда, как самому собрать такую команду и чему эти люди должны научиться, прежде чем ринуться в бой.

2.1. Что особенного в проектах, связанных с гибкой разработкой

Прежде чем перейти к описанию тонкостей работы гибкой команды, нужно прояснить некоторые общие моменты, касающиеся гибких проектов в целом.

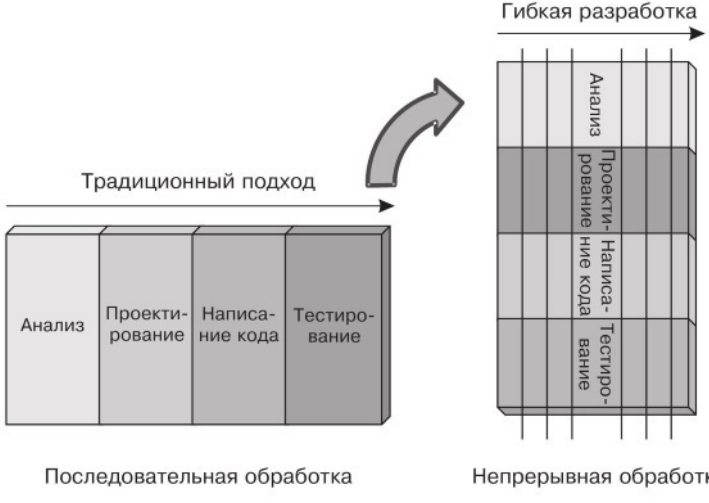
Первым делом отмечу, что в гибких проектах границы между ролями действительно размыты. Когда все идет хорошо, у человека, вливающегося в гибкую команду, возникает ощущение, что вся компания работает над маленьким стартапом. Люди принимаются за все сразу и делают все, что

может приблизить проект к цели, – независимо от роли или должности конкретного участника.

Разумеется, у всех сохраняются основные обязанности и люди обычно занимаются тем, в чем они особенно хороши. Но в гибком проекте такие узкоспециальные роли, как ана-

литик, программист и тестировщик, на самом деле не существуют – как минимум не существуют в традиционном понимании этих ролей.

Вторая деталь, специфичная для гибких команд, заключается в том, что анализ, проектирование, написание кода и тестирование идут постоянно, то есть не прекращаются.



Это означает, что все этапы работы перестают изолироваться друг от друга. Люди, выполняющие работу, должны быть объединены в единое целое и вместе ежедневно заниматься проектом.

Третий аспект, который нужно прояснить заранее, – на-

сколько важна для гибкости работы такая концепция одной команды и командной ответственности.

Качество выполнения гибкого проекта от начала до конца – задача всей команды. Отдела обеспечения качества (Quality Assurance, QA) нет, качество обеспечиваете вы сами, когда проводите анализ, пишете код или управляете проектом. Качество гарантируется на каждом шагу, поэтому в гибком проекте вы не услышите вопроса: «И как отдел гарантии качества проморгал эту ошибку?»

Итак, размытие ролей, постоянное сосредоточение на разработке и ответственность всей команды за все этапы проекта – вот что наверняка встретится вам при работе с гибкими командами.

Теперь давайте рассмотрим некоторые типичные дела, которыми занимаются гибкие команды. Это поможет нам самим успешно набирать такие команды.

2.2. Принципы действия гибкой команды

Прежде чем вы со своей командой начнете добиваться успеха, придется побороться за некоторые вещи, а также заточить команду для этих успехов.

Совместное размещение рабочих мест

Есть одна вещь, которая помогает радикально увеличить КПД вашей команды, – все должны работать в одном месте.

Команды, члены которых работают рядом, действительно показывают более высокие результаты. Ответы на вопросы находятся быстро. Проблемы решаются сразу же после их появления. Уменьшается количество трений между различными звеньями. Люди быстрее начинают доверять друг другу. С такой маленькой командой, работающей как единый организм, очень сложно соперничать.

Итак, если команды, работающие в одном помещении, так хороши, означает ли это, что удаленная команда не сможет выполнять гибкие проекты? Конечно, сможет.

Работа в удаленной команде стала для многих специалистов образом жизни. И хотя плотно сбитая, работающая в одном офисе команда всегда будет иметь некоторую фору перед удаленной, есть секреты, помогающие минимизировать это преимущество.

Например, в самом начале реализации проекта можно вы-

делить определенный бюджет на то, чтобы собирать разработчиков вместе. Даже если такая встреча продлится всего несколько дней (еще лучше, если удастся поработать в таком режиме пару недель), время, потраченное на знакомство друг с другом, привыкание к юмору коллег и совместные обеды, чрезвычайно помогает превратить разношерстную команду в крепкий, высокопроизводительный коллектив. Итак, постарайтесь для начала собрать всех вместе.

В конце концов, в вашем распоряжении масса технических средств (Skype, видеоконференции, социальные сети), помогающих превратить удаленную команду в группу, не уступающую по производительности работникам из маленького офиса.

Привлечение клиентов

В наше время все еще существует масса программ, которые пишутся командами разработчиков совершенно без участия клиентов. Это прискорбно и просто преступно.

Как у команды может получиться выдающийся, инновационный продукт, если сами люди, для которых он пишется, не участвуют в работе?

Заинтересованный клиент не пропускает презентаций, задает вопросы, реагирует на ход работы, направляет разработку и делится идеями, помогающими команде сделать нечто неординарное. Такие клиенты становятся ключевыми членами команды и полноправными коллегами разработчиков.

Стимулируйте незапланированное сотрудничество

В книге о компании «Пиксар» (The Pixar Touch) Стив Джобс рассказывает, как сильно успех фильмов этой компании зависел от незапланированного сотрудничества. После выпуска фильма «История игрушек-2» (Toy Story II), который чуть не поставил всю компанию на грань банкротства, руководство осознало, что сотрудники оказались слишком разобщены, изолированы друг от друга. Все могло закончиться крахом, если бы не собрали всех участников работы над фильмом вместе.

Именно для этого студия «Пиксар» приобрела участок 20 акров в Эмеривилле, штат Калифорния, и собрала всех сотрудников под одной крышей. Результат последовал незамедлительно. Контакты наладились, сотрудничество оптимизировалось, и коллеги смогли выпускать крупный фильм каждый год.

Вот почему гибкие методы (например, экстремальное программирование и скрам) предлагают бороться за участие клиента в работе. Это выражается в выделении особой роли

«*заказчик в команде*» (on-site customer) и в существовании в скраме специальной роли «*владелец продукта*» (product owner). Это очень важная работа. Подробнее все роли мы обсудим чуть позже.

Следующий принцип также проясняет, почему любой успешный гибкий проект требует участия клиента.

Принцип гибкой разработки

Бизнесмен и разработчик должны трудиться на протяжении проекта вместе, изо дня в день.

Вы можете спросить: «А что делать, если мой клиент не хочет участвовать в работе?» Возможно, клиент просто не может идти в ногу со временем или ему не особенно нужен этот проект, а быть может, он просто не считает, что вы движетесь к цели.

Какой бы ни была причина, вы должны завоевать определенное доверие клиента, это обязательно.

В следующий раз при встрече с клиентом скажите ему, что через две недели окончательно разберетесь с какой-то его проблемой.

Не просите разрешения. Не делайте из этого особенной церемонии. Просто возьмите какую-нибудь проблему или

досадную помеху и покончите с ней.

Затем сделайте так. Встретьтесь с клиентом через две недели, докажите ему, что проблема полностью исчерпана, и проделайте такую же операцию снова. Найдите другую проблему и заставьте ее исчезнуть.

Возможно, вам придется поступить так два или три раза, а то и больше, чтобы клиент стал интересоваться текущими проблемами. Просто знайте, что рано или поздно такой интерес придет.

Клиент начинает смотреть на вас иначе и понимать, кто вы такой: толковый специалист, с которым нужно считаться, чтобы дело спорилось.

Понимаете, может быть тысяча причин, по которым ваш клиент не участвует в работе. Вероятно, он устал от проектов, выполняемых в IT-отделе, или эта программа для него не так важна. Не исключено, что вы недостаточно подробно рассказали ему о том, как важна именно его роль для успеха всего проекта. А быть может, клиент на самом деле очень занят.

Я пытаюсь сказать, что, если вам нужно завоевать определенное доверие, начните понемногу наращивать его, и в итоге вы добьетесь своего.

Самоорганизация

Гибкая команда предпочитает, чтобы перед ней поставили цель, а затем не мешали всей команде разработать спо-

соб ее оптимального достижения. Для этого гибкая команда должна быть самоорганизующейся системой.

Самоорганизация заключается в том, что при необходимости нужно переступить через свое эго и вместе с командой понять, как вы, будучи командой (со всеми вашими индивидуальными навыками, пристрастиями и талантами), сможете выполнить конкретный проект максимально качественно.

«Разумеется, Бобби хорошо клепает код. Но у него еще особый талант к проектированию, он поможет нам сделать некоторые макеты».

«Да, Сьюзи – одна из лучших наших тестировщиц, но ее настоящий конек – выстраивание перед клиентом перспектив работы. У нее это отлично получается, и ей нравится это делать».

Это не означает, что разработчик должен быть экспертом по визуальному дизайну, а тестировщики – брать на себя управление проектом.

Скорее это признание того, что для создания оптимальной команды нужно исповедовать принцип «роль для человека, а не человек для роли».

Итак, как добиться самоорганизации своей команды?

♦ Команда допускается к планированию, оцениванию и может распоряжаться проектом.

♦ Вы не придаете особого значения ролям и их названиям, а сосредотачиваетесь на бесперебойном производстве

функциональных, протестированных программ.

♦ Вы ищете людей, способных брать на себя инициативу, то есть тех, кто сам прокладывает себе путь, а не сидит и не дожидается остальных.

Короче, вы отпускаете вожжи и позволяете сотрудникам самостоятельно делать свою работу.

Принцип гибкой разработки

Самая лучшая архитектура, требования и проекты рождаются в самоорганизующихся командах.

Теперь необходимо отметить, что самоорганизация как таковая, конечно же, очень хороша, но вся соль в том, к чему она приводит, – в расширении возможностей и индивидуальной ответственности.

Ответственная и полноправная

Хорошая гибкая команда всегда стремится отвечать за результаты своей работы. Эти люди знают, что клиент полагается на них и понимает, что без такой команды не обойтись. Поэтому члены гибкой команды никогда не увиливают от ответственности, которая неотделима от стремления достигать новых результатов с самого первого дня.

Разумеется, индивидуальная ответственность развивается только в командах, наделенных реальной полнотой полномочий. Давая команде право самостоятельно принимать решения и делать то, что она считает нужным, вы освобождаете пространство для инициативы и работы по собственному усмотрению. Люди начинают решать проблемы, не дожидаясь, пока им позволят это сделать.

На данном этапе никто не застрахован от ошибок. Но поверьте, игра стоит свеч.

Принцип гибкой разработки

Подбирайте для проекта мотивированных людей. Обеспечивайте им необходимую поддержку и хорошие условия труда и просто позвольте им сделать свою работу.

Конечно, создать ответственную и полноправную команду сложнее, чем просто сказать об этом. Не каждый готов к свободе действий. Зачем напрягаться, если можно просто прийти на работу, сесть и ждать, что тебе скажут.

Если вы чувствуете, что возникают проблемы с индивидуальной ответственностью, их легко решить – попросите команду продемонстрировать, как работает создаваемая программа.

Обычный прием, ставящий команду перед реальным клиентом, которому нужно показать, как выполняется поставленная задача, помогает кардинально повысить личную ответственность каждого члена.

Во-первых, команда осознает, что на нее и на результат ее работы рассчитывают реальные люди. Самые настоящие люди с насущными проблемами, для решения которых нужна заказанная программа.

Во-вторых, после первой же неудачной демонстрации для команды сразу же станет очень важно подготовить программу так, чтобы в следующий раз все работало. Люди сами будут брать на себя ответственность за решение подобных задач. Если этого не произойдет, значит, у вас большие проблемы.

Многофункциональность

Многофункциональной (cross-functional) называется команда, которая может реализовать требования клиента от начала и до конца. То есть команда должна обладать необходимым опытом и навыками и гарантировать, что сможет создать любую функцию, о которой попросит клиент.

Набирая людей в команду, ищите многостаночников, умеющих заниматься самыми разными делами. Говоря о таких программистах, я имею в виду людей, каждый из которых ориентируется во всем технологическом стеке (а не только в пользовательском интерфейсе или интерфейсе базы дан-

ных). Например, в случае тестировщиков и аналитиков это означает, что нам нужны люди, умеющие не только выполнять качественное тестирование, но и глубоко анализировать поставленные перед ними требования.

Специалисты привлекаются по мере необходимости, если команда не обладает каким-нибудь специфическим навыком (например, для настройки базы данных). Но в основном команда работает в тесном контакте на протяжении всего выполнения проекта.

Кто унес мой сыр?

«Кто унес мой сыр?» (Who Moved My Cheese? [Joh98]) – это бизнес-притча о мышах, которые проснулись однажды утром и обнаружили, что большой кусок сыра, вокруг которого им вольготно жилось, куда-то исчез. Его унесли. И теперь мыши в растерянности размышляли, что делать.

Кому-то переход к гибкой разработке может показаться таким отлучением от сыра.

Аналитик, переходящий к гибкому проекту, осознает, что этап анализа никогда не закончится.

Разработчик должен быть готов к тому, что ему придется писать тесты (и немало!).

То есть нужно понимать, что, предлагая людям работать в таком режиме, вы отбираете у кого-то сыр. И все, что вы можете для них сделать, – помочь им найти новый сыр (показать, как должны измениться их роли).

Разумеется, основным достоинством многофункциональ-

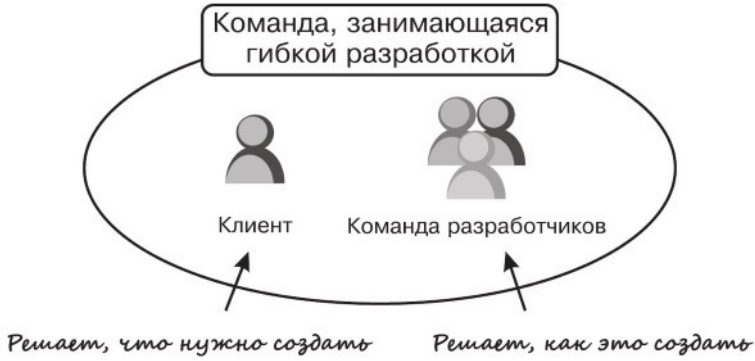
ной команды является скорость, с которой она способна работать. Ей не нужно ждать одобрения или договариваться о ресурсах с другими отделами, а можно с первого же дня начинать работу, не останавливая ее ни на день.

Конечно, вы должны обрисовать людям, чего от них ожидаете, и рассказать о результатах, которых хотите добиться, набирая свою команду.

А теперь поговорим о ролях.

2.3. Роли, которые встречаются в типичной команде

Гибкие методы, такие как скрам и экстремальное программирование, предусматривают при реализации проекта совсем немного формальных ролей. Есть люди, которые знают, что должно быть сделано (клиенты), и люди, знающие, как это сделать (команда разработчиков).



Если у вас возникает вопрос: «А где все программисты, тестировщики и аналитики?» – не волнуйтесь, они никуда не исчезли. Просто при гибкой разработке не так важно, кто именно играет конкретные роли.

Начнем с рассмотрения самой важной роли в гибком проекте – клиента.

Клиент гибкого проекта

«Источником истины» здесь является клиент, сотрудничающий с командой разработчиков, – от него поступают все требования к выполняемому проекту. Ведь именно для клиента пишется программа.

В идеале клиент должен ориентироваться в теме проекта. Это человек, глубоко погруженный в дело, ему действительно небезразлично, что делает программа, как она выглядит и функционирует. Кроме того, он с энтузиазмом направляет команду, отвечает на вопросы, словом, проявляет отдачу.

Кроме того, клиент устанавливает приоритеты. Он реша-

ет, что нужно создать и когда.

Все это не происходит в вакууме. Речь идет о совместной работе с командой разработчиков, ведь могут быть технические причины, по которым целесообразно сделать сначала одни элементы, а потом другие (иными словами, снизить технологический риск).

Однако обычно приоритеты расставляются с точки зрения бизнеса, а затем начинается работа с командой, которой нужно выполнить намеченный план, чтобы все получилось.

И именно клиентам приходится принимать решения относительно того, без чего можно обойтись, если поджимают сроки и начинают заканчиваться деньги.

Разумеется, чтобы все это получилось, требуется очень тесное сотрудничество клиента с командой (в идеале – в течение всего рабочего дня). В ранних версиях экстремального программирования было принято говорить о «заказчике в команде» (on-site customer). В скраме аналогичная роль называется «владелец продукта» (product owner).

Но не переживайте, если не удается заполучить клиента на полный рабочий день, – это получается у мизерной доли команд. Даже без заказчика в команде вы можете сделать гибкий и весьма успешный проект. Не на всех проектах вообще требуется такой участник.

Еще важнее чувствовать тот дух, на котором строятся гибкие методы, подобные экстремальному программированию или скраму. Сущность этих методов заключается в том, что

чем более непосредственный контакт у вас с клиентом, тем лучше.

Итак, добейтесь настолько тесного сотрудничества с клиентом, насколько это возможно. Убедитесь, что клиент понимает важность своей роли. Нужно, чтобы клиент был полностью свободен в своих действиях и сам хотел принимать решения, необходимые для успешного завершения проекта.

Теперь поговорим о команде разработчиков.

Команда разработчиков

Гибкая команда разработчиков – это группа многофункциональных специалистов, которые могут работать с любой функцией, которую желает получить клиент, и превратить ее в готовый, рабочий программный компонент. В состав команды входят аналитики, разработчики, тестировщики, ад-

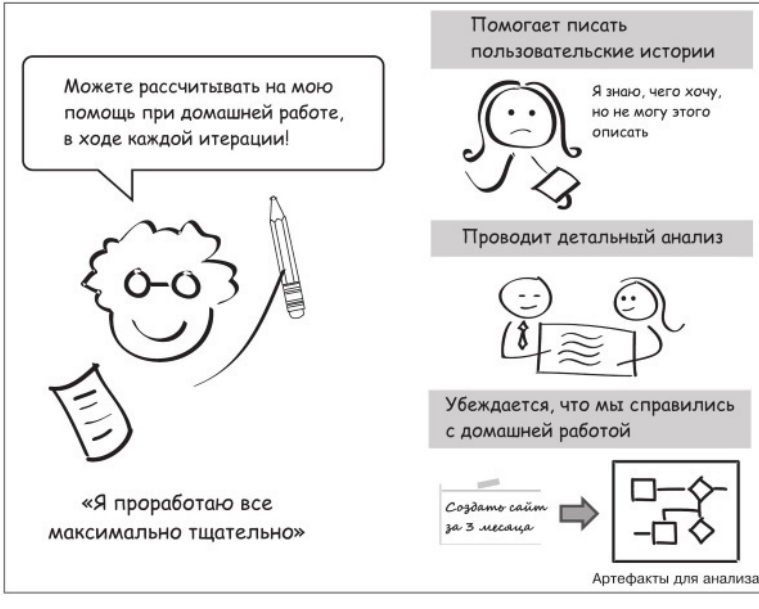
министраторы баз данных и все остальные специалисты, которые нужны для превращения пользовательских историй в программу, готовую для реального использования.

Как бы я ни обожал тот дух и те принципы, на основе которых работает гибкая команда, не имеющая формально распределенных ролей, должен признаться, что попытка пригласить глубоко консервативную команду разработчиков и сообщить ее членам, что им нужно «самоорганизоваться», на практике у меня никогда не срабатывала.

Чтобы чувствовать себя уверенно, нужна точность формулировок. Надо сразу ясно сказать, что в гибкой команде границы между отдельными ролями размыты и что от каждого участника требуется умение сражаться на нескольких фронтах. Но мне лучше удавалось придавать коллективам нужный вид, если я объяснял им гибкую методологию в понятных им выражениях.

Если ваша команда именно такая, то обратите внимание на следующие описания гибких ролей, которые помогут людям адаптироваться к новым условиям и понять, как изменятся их роли в гибком проекте.

Гибкий аналитик



Когда мы приступаем к разработке определенной функции, кто-то должен досконально с ней разобраться и описать все тончайшие детали ее работы. Это задача нашего гибкого аналитика.

Аналитик похож на неутомимого сыщика, задающего глубокие зондирующие вопросы и при этом испытывающего кайф от тесного сотрудничества с клиентом. В ходе совместной работы они пытаются понять, что же нужно от программы.

Аналитик выполняет в гибком проекте множество задач:

помогает клиенту писать пользовательские истории (см. главу 6); выполняет глубокий анализ, когда дело доходит до разработки; помогает создавать имитационные объекты (mock-ups) и прототипы; использует в ходе анализа все доступные ему инструменты, чтобы донести до разработчика сущность пользовательских историй.

Подробнее о функционировании гибкого анализа мы поговорим в разделе 9.4.

Гибкий программист

Пока не написан код, все сводится только к конструктивным намерениям. Написанием кода занимаются наши гибкие программисты.

Гибкий программист – это профессионал, так как он очень серьезно относится к качеству программы. Лучшие из таких программистов одновременно являются увлеченными тестировщиками, гордящимися своей работой и всегда старающимися написать максимально качественный код.

Поэтому существуют определенные вещи, без которых не обойтись программистам, желающим регулярно создавать отличные многофункциональные продукты.

♦ Они пишут много тестов и часто пользуются ими при разработке (см. главы 12 и 14).

♦ Они постоянно дорабатывают и оптимизируют архитектуру программы по мере работы (см. главу 13).

♦ Они уверены, что базовый код всегда готов к использованию и может быть развернут по первому требованию (см. главу 15).

Программист также тесно сотрудничает с клиентом и другими членами команды и гарантирует, что написанный код работает, что он максимально прост и что внедрение программы для практического использования не составит никакого труда.

Гибкий тестировщик

Гибкие тестировщики знают, что одно дело – написать продукт, а другое – гарантировать, что он работает. Поэтому такой тестировщик подключается к проекту уже на ранних этапах, заблаговременно обеспечивая успешное выполнение пользовательских историй и то, что программа, запущенная в практическую работу, действительно не подведет.

Поскольку в гибком проекте тестировать нужно буквально все, ни один этап работы не обойдется без участия тестировщика. Он будет работать бок о бок с клиентом, помогая

ему оформить предъявляемые требования в виде тестов.

Что, если начинать каждый проект вот так?

Предположим, в начале каждого проекта вы вместе с командой пытаетесь ответить на четыре простых вопроса о себе.

- В чем я особенно хорош?
- Как я привык работать?
- Что я ценю?
- Каких результатов можно от меня ожидать?

Затем, получив новые идеи, задайте те же вопросы коллегам, чтобы они рассказали вам, в чем они хороши, как работают, что ценят и какой результат могут выдать.

Эта идея называется упражнением Друкера[4]. При всей простоте оно очень помогает сплотить команду, наладить необходимую коммуникацию и уровень доверия, необходимый для любой высокопроизводительной команды.

Тестировщик будет тесно сотрудничать с разработчиками, помогая автоматизировать тесты, отыскивая «дырки» и выполняя широкое исследовательское тестирование, пытаясь проверить приложение под любым возможным углом.

Кроме того, тестировщик будет представлять себе целостную картину тестирования и никогда не упустит из виду тестирования с возрастающей нагрузкой, масштабируемости и всех остальных деталей, которые требуется учесть для создания высококлассного продукта.

---

[4] http://agilewarrior.wordpress.com/2009/11/27/the-drucker-exercise.

В книге Джанет Грегори и Лизы Криспин *Agile Testing: A Practical Guide for Testers and Agile Teams* [GC09] подробно описана важность роли тестировщика при гибкой разработке.

Подробнее о механизмах гибкого тестирования мы поговорим в разделе 9.6.

Гибкий менеджер проектов

Гибкий менеджер проекта знает, что успех невозможен без плодотворной работы всей команды. Вот почему хороший менеджер проекта сделает все возможное и невозможное, чтобы ничто не мешало его команде достичь успеха.

В частности, он займется планированием и перепланированием, а при необходимости – корректировкой курса (см. главу 8).

Кроме того, к его задачам относится улавливание ожиданий всех людей, занятых в работе над проектом: передача отчетов о состоянии дел владельцам (stakeholders), стимулирование взаимодействий внутри компании, а также защита рабочей группы от неблагоприятных внешних воздействий. Все это – обязанности хорошего гибкого менеджера проектов.

Хороший менеджер проекта не рассказывает команде, чем ей заниматься, – этого просто не требуется. Он помогает создать такую среду, в которой команда будет чувствовать себя максимально независимо и продолжать отлично работать и в отсутствие менеджера проектов. На самом деле фирменной чертой хорошего гибкого менеджера проектов является умение исчезнуть и вернуться через две недели, но так, чтобы никто этого не заметил.

Подробнее об управлении проектами мы поговорим в главах 8 и 9.

Гибкий разработчик взаимодействия с пользователем

Разработчики уровня взаимодействия пользователя с программой сосредоточены на создании полезных, удобных и приятных функций, обеспечивающих такое взаимодействие. Специалист, интересующийся проблемой удобства работы с программой (юзабилити), будет стремиться понять, что нужно пользователю, а затем сотрудничать с оставшейся командой, чтобы найти наилучшие способы удовлетворения

пользовательских потребностей.

К счастью, многие практические методы, используемые юзабилити-экспертами, хорошо согласуются с духом гибкой разработки программ. Акцент на ценности продукта, быстрая коммуникация и максимальное удовлетворение потребностей клиента – общие цели как гибких разработчиков, так и специалистов по удобству программы.

Кстати, юзабилити-экспертам не в новинку работать постепенно, и они привычны к итерациям. Они проектируют и создают новые функции по мере того, как пишется код (а не пытаются сделать все заранее и кардинально опередить всю команду).

Если вы можете заполучить в ваш проект специалиста по юзабилити, считайте, что вам повезло. Такой человек может поделиться массой полезного опыта и знаний и очень поможет общему делу в области анализа и разработки уровня взаимодействия с пользователем.

Все остальные

Перечислю остальные важные роли и специалистов, которых не упомянул выше. Это администраторы баз данных, системные администраторы, технические писатели, инструкторы, специалисты по улучшению текущей деятельности, инфраструктуры и обслуживанию сетей. Все они – члены команды разработки и равноправные участники проекта.

В скраме есть роль руководителя (scrum master). В гиб-

ком проекте ему отводится место тренера и продюсера рок-звезды в одном флаконе. Такие тренеры могут быть очень полезны при «раскочегаривании» новых команд. Они умеют объяснить и привить принципы гибкой разработки и соответствующую философию, а также гарантировать, что команда не собьется с курса и не вернется к прежним вредным привычкам. Работа тренера хорошо описана в книге Agile Coaching [SD09].

Опытной команде, как правило, не требуется специальный тренер, но новому проекту такой специалист определенно не повредит.

И последнее: рассказывая об этих ролях, дайте людям понять, что в гибком проекте вполне нормально (и ожидаемо), что человек будет одновременно играть несколько ролей.

Иными словами, пусть аналитик знает, что разработчик имеет полное право беседовать с клиентом (это даже приветствуется). Пусть тестировщики не удивляются тому, что разработчику придется написать немало автоматизированных тестов. А если в вашем проекте не будет специального разработчика пользовательских интерфейсов, это не означает, что никто не собирается заниматься юзабилити и дизайном. Разумеется, все это будет сделано, только другими людьми, которые исполняют в гибком проекте сразу несколько ролей.

В завершение немного поговорим о том, как набирать людей в команду.

2.4. Советы по подбору команды для гибкой разработки

Хотя большинству людей должно понравиться работать в высокопроизводительной гибкой команде, есть некоторые вещи, на которые нужно обращать внимание при подборе квалифицированных профессионалов.

Ищите универсалов

Универсалы хорошо приживаются в гибких проектах, поскольку сама методология требует от людей систематически работать и использовать для этого все предоставляемые возможности. Если говорить о программисте, то нам нужен разработчик, который имеет представление обо всем технологическом стеке проекта. Аналитики и тестировщики должны соответственно проводить анализ и уметь работать с тестами.

Кроме того, универсал легко вживается в разные роли. Сегодня человек пишет код, завтра занимается анализом, а послезавтра – тестированием.

Люди, не боящиеся неопределенности

Если проект гибкий, это не означает, что в нем все пойдет как по маслу. Требования не будут преподноситься на блюдечке – нужно работать и самостоятельно выяснять их.

Планы будут меняться, и вам придется приспосабливаться к этим изменениям.

Ищите людей, которые не боятся отбивать закрученные мячи, умеют держать удар и при необходимости изменять курс, не прекращая движения вперед.

Командные игроки, умеющие подавлять свое эго

Звучит избито, но для гибкой разработки лучше подойдут люди, которые умеют работать слаженно и подавлять собственное эго.

Не всем по вкусу расплывчатость ролей, принятая в гибкой методологии. Некоторые люди стараются защищать область, которую считают своей епархией.

Просто ищите тех, кто не имеет противоречий с самим собой, не боится делиться знаниями и искренне наслаждается взаимным обучением и ростом.

Мастер-сэнсэй и воин, постигающий искусство

УЧЕНИК: *Мастер, я запутался. Если в гибком проекте нет предопределенных ролей, то как же он работает?*

МАСТЕР: *Команда сделает то, что должно быть сделано.*

УЧЕНИК: *О да, Мастер, но если нет специальной роли тестировщика, как можно быть уверенным, что все необходимые тесты будут проведены?*

МАСТЕР: *Без тестирования никак не обойтись, поэтому команда будет им заниматься. Команда решает, сколько нужно тестов и сколько сил на это потратить.*

УЧЕНИК: *А если никто не хочет заниматься тестированием? Что, если всем нравится просто сидеть и писать код?*

МАСТЕР: *Тогда нужно найти людей, которым нравится тестировать, и самому убедиться, какими ценными членами твоей команды они станут.*

УЧЕНИК: *Благодарю тебя, Мастер. Я подумаю над этим.*

Что дальше?

Мы рассмотрели, как в гибких проектах исчезают четкие границы между ролями, почему команда будет работать наиболее успешно, если все соберутся в одном месте, и как, подыскивая людей в команду, находить специалистов-универсалов и тех, кто не боится неопределенности.

Теперь мы готовы сделать, пожалуй, один из важнейших шагов, чтобы отправить наш гибкий проект в свободное плавание. Поговорим об этапе, о котором почти ничего не сказано в большинстве гибких методологий, – как зарождается гибкий проект.

Из второй части книги вы узнаете, как с самого начала сориентировать свой проект на путь к успеху и гарантировать, что вы подобрали для работы нужных людей.

Часть II
Концептуализация проекта при гибкой разработке

Глава 3
Главное – никого не забыть

Многие проекты умирают в зачаточном состоянии. Обычно это происходит по одной из следующих причин:

- ♦ неумение задавать правильные вопросы;
- ♦ боязнь задавать сложные вопросы.

В этой части мы поговорим о мощной методике построения перспектив, которую условно назовем стартовой колодой (inception deck). Она помогает найти ответы на 10 вопросов, без которых лучше не начинать какой-либо софтверный проект. Испытав команду на данном этапе, вы узнаете, все ли нужные люди подобраны для проекта и в правильном ли направлении вы движетесь. Это произойдет еще до написания самой первой строки кода.

3.1. Из-за чего погибает большинство проектов

В начале любого нового проекта люди обычно имеют поразительно разные представления об общей цели.

Для проектов это может быть губительно. Ведь, хотя мы и описываем наше видение общего дела на одном и том же языке, стоит нам приступить к работе – и мы понимаем, что

думали о совершенно разных вещах.

И проблема не в том, что нам не удалось прийти к общему мнению уже на старте (это естественно). Проблема в том, что проекты начинаются *еще до того*, как найдены все нужные люди.

Ошибочное мнение о том, что консенсус достигнут там, где его нет и в помине, губит большинство проектов.

Нам нужно сформулировать план, который:

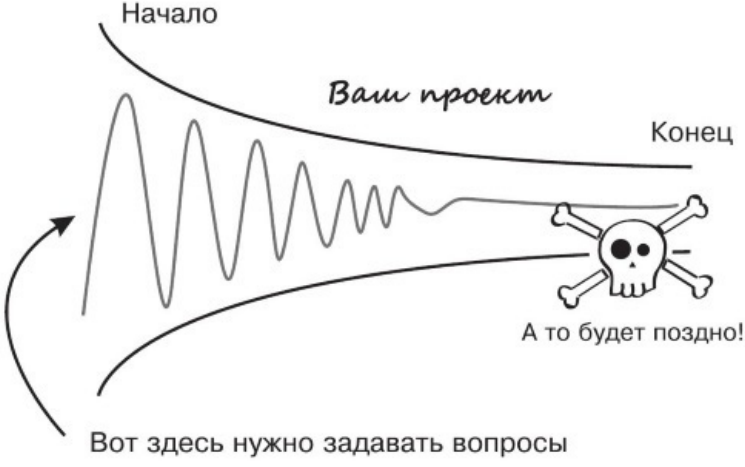
♦ позволяет сообщить команде цели, суть проблемы и контекст, в котором реализуется проект, так, чтобы при работе сотрудники могли принимать осознанные решения;

♦ предоставляет владельцам информацию, помогающую им решить, браться или не браться за дело, начинать проект или нет.

Единственный способ выстроить такой план – не бояться задавать вопросы.

3.2. Не избегайте сложных вопросов

Когда я работал в Новой Зеландии, мне представилась возможность сопровождать в поездке одного из крупнейших специалистов по маркетингу из компании ThoughtWorks – джентльмена по имени Кейт Доддс. Одна из многих вещей, которым научил меня Кейт, заключается в том, как важно уметь задавать самые сложные вопросы в самом начале любого нового предприятия.



Как видите, начиная любое новое дело (то есть проект), вы имеете большой простор для постановки вопросов и при

этом ничего не теряете. Можно задавать общие вопросы, например, как приведенные ниже.

- ♦ Насколько опытна ваша команда?
- ♦ Занимались ли вы такими вещами ранее?
- ♦ Сколько денег у нас в распоряжении?
- ♦ Кто отдает приказы в этом проекте?
- ♦ Не смущает ли вас ситуация, когда в проекте участвуют два аналитика и тридцать разработчиков?
- ♦ Перечислите проекты, для работы над которыми вам пришлось пригласить команду молодых специалистов, практически незнакомых с объектно-ориентированным программированием, и успешно переделать устаревшую систему мейнфрейма для работы с Ruby on Rails – и сделать все это с помощью гибкой методологии.

Такие же вопросы необходимо задавать при запуске гибкого проекта. Нужно прояснить все щекотливые моменты с самого начала. Это нужно делать на этапе концептуализации проекта.

3.3. Знакомство со стартовой колодой

На этом уровне мы словно включаем прожектор, рассеивающий неясность и таинственность, окружающие ваш гибкий проект. Здесь мы прорабатываем 10 сложных вопросов и ситуаций, которые просто необходимо разрешить еще до начала проекта.

В компании ThoughtWorks такой подход часто используется для анализа той части первичных работ над проектом, о

которой практически ничего не говорится ни в экстремальном программировании, ни в скраме. Речь идет о *закладке проекта* (project chartering). Мы знали, что глубокий шестимесячный анализ и упражнения в сборе требований нам не подходят, но не могли придумать какую-нибудь более легковесную альтернативу. И именно в таких условиях Робину Гиббонсу пришла в голову мысль о стартовой колоде: быстром, легком способе извлечения из проекта самой его сути и рассказа об общем понимании задачи всей команде и другим участникам проекта.

3.4. Как это работает

Смысл стартовой колоды заключается в том, что, если нам удастся собрать нужных людей в одной комнате и задать им правильные вопросы, это будет невероятно полезно для формулирования наших общих ожиданий, которые мы связываем с проектом.

Проведя в команде несколько упражнений и собрав их результаты в виде презентации (обычно – PowerPoint), можно совместно выработать достаточно хорошее понимание того, в чем заключается проект, чем он не является и что потребуется для его реализации.

В составлении стартовой колоды должны участвовать люди, непосредственно связанные с реализацией проекта. Речь идет о клиентах, владельцах, членах команды, разработчи-

ках, тестировщиках, аналитиках – всех, кто способен сделать реальный вклад в эффективное выполнение проекта.

Отдельно подчеркну, как важно, чтобы в этом проекте участвовали владельцы. Ведь стартовая колода – это инструмент, предназначенный не только для нас, но и для них, и с ее помощью они могут принять решение о том, стоит ли вообще начинать проект.

На построение типичной стартовой колоды уходит от пары дней до примерно двух недель. Она равноценна примерно шести месяцам работ по планированию проекта и должна пересматриваться при внесении любых крупных изменений в суть или направление развития проекта.

Такой пересмотр необходим, поскольку стартовая колода – это живая, открытая система. Она не из тех вещей, которые можно один раз сделать и положить на полку. До самого завершения проекта схема должна висеть в офисе, где трудится команда, и напоминать о том, над чем идет работа и с какой целью.

Разумеется, вопросы и упражнения, представленные здесь, – это далеко не все. До начала проекта вам придется обдумывать и другие вопросы, разрабатывать упражнения и прояснять детали.

Итак, используйте такую схему в качестве отправной точки, но не следуйте ей слепо и не бойтесь корректировать ее под себя.

3.5. Сущность стартовой колоды

Ниже перечислены основные вопросы и упражнения, прорабатываемые на этапе концептуализации проекта.

1. **Зачем мы здесь собрались?** Это быстрое напоминание о нашей цели, наших клиентах, а также о том, почему мы решили заняться данным проектом в первую очередь.

2. **Составление блицрезюме**. Если бы у нас было 30 секунд, за которые нужно описать наш проект в двух фразах, что бы мы о нем сказали?

3. **Разработка оформления продукта**. Если бы мы быстро листали журнал и наткнулись на рекламу нашего продукта или услуги, то что бы она нам сообщила, и еще важнее – согласились ли бы мы за это заплатить?

4. **Составление списка того, что мы не собираемся делать**. Вполне ясно, что мы собираемся делать при реализации нашего проекта. Давайте еще точнее опишем ситуацию и подчеркнем, чего мы ни в коем случае делать не будем.

5. **Встреча с коллегами**. Сообщество специалистов, занятых в проекте, всегда больше, чем кажется. Почему бы не пригласить их на кофе, чтобы все могли познакомиться друг с другом?

6. **Демонстрация решения**. Давайте нарисуем общий концептуальный проект технической архитектуры, чтобы убедиться, что все мы одинаково представляем себе пред-

стоящую работу.

7. **Что не дает нам покоя?** Иногда в ходе выполнения проектов происходят неприятные вещи. Но если поговорить о них и подумать, как их избежать, то, возможно, все будет не так плохо.

8. **Определение временных параметров**. Сколько времени займет проект: три месяца, а может, шесть или девять?

9. **Определение требуемого результата.** Проекты зависят от таких факторов, как время проекта, его функционал, бюджет и качество. Что наиболее и наименее важно для данного проекта в настоящий момент?

10. **Что нам требуется для достижения результата?** Сколько времени будет длиться проект? Сколько он будет стоить? И какая команда нам нужна для реализации этого проекта?

Мы изучим стартовую колоду в два этапа. В главе 4 поговорим о том, зачем мы беремся за проект, а в главе 5 рассмотрим, как его реализовать.

Глава 4
Общее представление о ситуации

Разработка программного обеспечения – один из уникальных видов деятельности, в которой сочетаются черты проектирования, конструирования, искусства и науки.

Ежедневно команды должны принимать тысячи решений и идти на не меньшее количество компромиссов. А без верного контекста и общего представления о ситуации такие ре-

шения не могут быть полностью осознанными или взвешенными.

При изучении первой части стартовой колоды мы должны досконально выяснить, на чем основан наш проект.

- ♦ Для этого нужно ответить на ряд вопросов.
- ♦ Для чего мы здесь собрались?
- ♦ Каково блицрезюме нашего проекта?
- ♦ Как будет выглядеть реклама нашего продукта?
- ♦ Чего мы не собираемся делать?
- ♦ Кто работает с нами в команде?

В финале этой главы вы и ваша команда будете ясно понимать, какова цель проекта, что именно вы создаете и почему вы это делаете. Обо всем этом вы сможете с легкостью рассказать другим.

Но сначала зададим нашим спонсорам вопрос, зачем мы здесь?

4.1. Вопрос: зачем мы здесь?

Зачем мы здесь?

Чтобы спокойно отслеживать, какие работы идут на нашей виртуальной стройке

Прежде чем любая команда сможет добиться успеха, она должна понять, зачем она занимается решением конкретной задачи. Поняв это, команда получает возможность:

- ♦ принимать оптимальные и осознанные решения;
- ♦ лучше выполнять работу, уравновешивая противодействующие силы и идя на компромиссы;
- ♦ выдавать более инновационные и качественные решения, поскольку команде разрешено думать самостоятельно.

Задача сводится к тому, чтобы понять намерение начальника и сформулировать для себя, как достичь поставленной цели.

«Тойота»: компания, умеющая видеть и достигать

В замечательной книге The Toyota Way [Lik04] Джеффри Лайкер рассказывает историю о том, как однажды в 2004 году главному инженеру этой компании поручили переработать модель «Тойота-Сиенна» для североамериканского рынка. Чтобы почувствовать, как жители Северной Америки живут, работают и занимаются своими машинами, он с командой проехал на «Тойоте-Сиенна» по всем штатам США и Мексики, а также по всем провинциям Канады.

И вот что выяснилось.

• Североамериканские водители больше едят и пьют в машине, чем японские автомобилисты (так как в Японии обычно приходится ездить на более короткие расстояния). Поэтому в любой «Тойоте-Сиенна» есть центральный лоток и 14 подставок для чашек.

• На канадских дорогах более высокий поперечный уклон, чем в США (выгнутый в середине), поэтому при вождении очень важно контролировать скольжение.

• Сильные ветры, дующие в провинции Онтарио, требуют особого внимания к устойчивости автомобиля к боковому ветру. Если вы поедете куда-нибудь, где сильные боковые ветры, вас приятно удивит устойчивость и легкость в движении новой «Тойоты-

Сиенна».

• Если бы главный инженер мог всего лишь прочитать об этих проблемах в маркетинговом отчете, он не почувствовал и не осознал бы на собственном опыте суть данных проблем.

Идите и попробуйте сами

Одно дело – догадываться, зачем мы здесь, и совсем другое – знать об этом с определенностью. Чтобы действительно увидеть проблему с точки зрения клиента и понять, что ему нужно, требуется поставить себя на его место.

Пойти и посмотреть – означает растормошить команду и познакомить ее с той сферой, где будет происходить действие.

Например, если вы создаете систему обеспечения производственной безопасности для инженерной компании, которую предполагается использовать на промплощадке, – отправьтесь на место разработок. Поговорите с офицерами службы безопасности. Посмотрите на тягачи. Познакомьтесь со стесненными условиями, неустойчивым интернет-соединением, а также с теми маленькими кабинетами, в которых будут работать ваши клиенты. Проведите день на этом месте и поработайте с людьми, которым ежедневно придется пользоваться системой, которую вы собираетесь разрабатывать.

Войдите в курс дела, задавайте вопросы и ненадолго превратитесь в своего клиента.

Как понять заказчика

Заказчик выражает свои намерения в краткой фразе, пожелании или формуле, которая обобщает цели и задачи вашего проекта. Эта краткая формулировка должна служить путеводной звездой, на которую можно взглянуть в последнюю минуту, в самом разгаре битвы и решить, атаковать или отступить.

В книге Made to stick [HH07] Чип и Дэн Хиты рассказывают, как в компании Southwest Airlines обсуждали, включить ли в меню одного из рейсов куриный салат «Цезарь».

Когда был задан вопрос, позволит ли это снизить стоимость билета (этого хотел добиться главный исполнительный директор Хербс Келлехер), стало понятно, что добавлять куриный салат абсолютно бессмысленно.

Итак, желание заказчика в начале проекта не обязательно должно сводиться к чему-то большому и пафосному. Оно может быть очень простым и совершенно не выходить за рамки вашего проекта.

Суть этого упражнения – заставить людей сказать о том, что у них на уме, а затем уточнить у клиента, действительно ли это – все, что требуется.

4.2. Создание блицрезюме

Блицрезюме

- Для [руководителей строительных работ],
- которые [должны отслеживать, какие работы проводятся на промплощадке],
- обеспечивают [соблюдение техники безопасности при проведении работ]
- в [системе выполнения требований техники безопасности]
- которая [создает, отслеживает и проверяет разрешения на выполнение работ].
- В отличие от [современных бумажных систем документооборота]
- наш продукт [основан на веб-технологиях, поэтому к нему можно обратиться когда угодно и откуда угодно].

10 причин взяться за выполнение вашего проекта

Недавно я выполнял это упражнение с командой, перед которой была поставлена задача создания инвойсов для нового отдела компании. Меня удивил разброс мнений, царивший в команде относительно того, зачем начался этот проект.

Некоторые считали, что смысл – снизить количество страниц в стандартном инвойсе и сэкономить таким образом бумагу. Другие думали, что так упростится заполнение инвойса и поэтому снизится нагрузка на колл-центр. Третьи полагали, что так реализуется возможность запуска целевых маркетинговых кампаний, задача которых – повысить продажи товаров и услуг.

Все это хорошие ответы, но ни один из них не оправдывал проект сам по себе. Понадобилось

немало дискуссий и дебатов, чтобы перед командой вырисовалась истинная цель проекта и пришло ее понимание. На самом деле все делалось именно для упрощения инвойса и снижения нагрузки на колл-центр.

Поторопитесь! Венчурный инвестор, которого вы пытались выловить для разговора тет-а-тет на протяжении трех месяцев, только что вошел в лифт, и у вас есть 30 секунд, чтобы познакомить этого капиталиста с идеей вашего новоиспеченного проекта. Сможете – ваше предприятие получит столь необходимую поддержку. Не сможете – тогда снова придется питаться одним роллтоном.

Блицрезюме – это и есть ваша «речь в лифте». В нем вы должны сформулировать свою идею за крайне короткий промежуток времени. Блицрезюме предназначены не только для завлечения рисковых предпринимателей-авантюристов. В форме такого резюме еще удобно кратко и точно описывать новые софтверные проекты.

Хорошее блицрезюме помогает решить ряд очень важных для проекта задач.

1. **Вносит ясность.** Блицрезюме – это не попытка удовлетворить «и наших и ваших». Оно заставляет команду отвечать на сложные вопросы о том, чем является будущий продукт и для кого он предназначен.

2. **Заставляет команду задуматься о клиенте.** Делая

акцент на том, что будет делать программа и как именно, команды приобретают ценные идеи, касающиеся особенностей данного продукта и того, почему клиент купит в первую очередь именно его.

3. **Помогает вникнуть в суть дела.** Блицрезюме, подобно лазеру, прорезает массу хлама и попадает в самое сердце проекта. Такая ясность помогает расстановке приоритетов и значительно улучшает соотношение «сигнал – помеха», то есть количество того, что действительно важно.

Теперь давайте рассмотрим шаблон для составления блицрезюме.

Шаблон блицрезюме

- Для [адресат сообщения],
- который [утверждение о необходимости или о предоставляющейся возможности]
- это [название продукта]
- относится к [категория продукта]
- и при этом [основная выгода, убедительная причина приобрести].
- В отличие от [основное конкурентное предложение]
- наш продукт [объяснение основного положительного отличия].

Существует несколько способов преподнесения блицрезюме. Тот, который предпочитаю я, взят из книги Джеффри Мура *Crossing the Chasm* [Moo91].

♦ *Для* [адресат сообщения]. Объясняется, на кого ориентирован проект или кому он мог бы принести пользу.

♦ *Который* [утверждение о необходимости или о предоставляющейся возможности]. Расширенное представление проблемы или потребности, стоящей перед клиентом.

♦ *Это* [название продукта]. Проект начинается с присвоения имени. Название важно, так как оно помогает понять ваши намерения.

♦ *Относится* к [категория продукта]. Объясняется, чем, по сути, является данный продукт или услуга и какова полезная нагрузка проекта.

Быть кратким нелегко

Одна из причин, по которой блицрезюме оказывает такое сильное воздействие, – его краткость. Но не думайте, что написать короткое резюме так просто.

Вам и вашей команде потребуется немало времени для написания хорошей речи, поэтому не беспокойтесь, если все получится не сразу. Создать хорошее блицрезюме бывает непросто, но оно стоит затраченных усилий.

«Я написал бы вам еще короче, но у меня нет на это времени». – Блез Паскаль, «Письма к провинциалу», XVI.

♦ *И при этом* [основная выгода, убедительная причина приобрести]. Объясняется, почему клиент не отказался бы купить в первую очередь именно этот продукт.

♦ *В отличие от* [основное конкурентное предложение]. Говорится, зачем нужен продукт, если на рынке уже есть аналог.

♦ *Наш продукт* [объяснение основного положительного отличия]. Здесь мы помогаем почувствовать разницу и объясняем, что в нашем предложении особенного, чем оно лучше альтернатив, предлагаемых конкурентами. Это самое важное. Именно на данном этапе мы объясняем, почему стоит вложить деньги в наш проект.

♦ В двух красивых фразах, которые вы успеете сказать в лифте, должна быть заключена суть проекта или идеи. Необходимо рассказать, что собой представляет проект, для чего он нужен и почему стоит купить именно его.

Есть несколько способов составить вместе с командой такое блицрезюме. Можно напечатать шаблон, попросить каждого попробовать заполнить его самостоятельно, прежде чем собирать всех вместе.

Или, если вы хотите спасти елочку и не тратить бумагу, просто выведите шаблон на экран с помощью проектора и попробуйте заполнить его вместе со всей группой, по одному разделу за раз.

Когда блицрезюме будет готово, следует проявить творческие способности и перейти к созданию оформления для продукта.

4.3. Разработка оформления продукции

Система обеспечения
производственной безопасности
для инженерной компании

Идеальный вариант
для горнодобывающей промышленности

Обрабатывайте допуски к работе быстрее!
Обрабатывайте допуски к работе надежнее!
Отслеживайте рабочее время лучше!

Где вам это потребуется.
Когда вам это потребуется

Иногда компьютерные программы оказываются для компаний необходимым злом. Чтобы не принимать на себя весь риск и все неопределенные моменты, связанные с крупными проектами, многие клиенты предпочтут отправиться в местный супермаркет, вытащить карточку и просто закупить все, что надо.

Возможно, мы еще долго не увидим полок супермаркетов, уставленных программными продуктами, которые заботливо упакованы в пластиковые коробки и стóят шестизначные суммы. Но в связи с этим возникает интересный вопрос. Если бы можно было купить программу в супермаркете, как бы выглядела ее упаковка? И – еще важнее – купили бы мы ее?

Создавая оформление для своего проекта и задавая вопрос, почему клиент должен им заинтересоваться и купить его, вы сосредотачиваете внимание вашей команды на том, что более всего интересует вашего потребителя, и на основных преимуществах вашего продукта. При работе команда должна четко знать ответ и на первый, и на второй вопрос.

Как это работает

Представляю, о чем вы сейчас думаете. «Какой из меня креативщик. Я не силен в рекламе. Конечно, я не смогу разрекламировать наш продукт». Позвольте вас заверить – сможете! Я покажу вам, как это делается, за три простых этапа.

Этап 1. Мозговой штурм о достоинствах вашего проекта

Никогда не рассказывайте клиенту о характерных особенностях вашего продукта – это неважно. Людей интересует то, как ваш продукт сможет облегчить им жизнь, иными словами, какая от него польза.

Предположим, мы пытаемся убедить семью покупателей в том, что ей очень пригодился бы мини-вэн. Мы могли бы дать им подробное описание машины либо остановиться на том, как такой автомобиль полезен в обычной жизни.

Функции	Польза
Двигатель 245 лошадиных сил	Легкий ход машины по шоссе
Круиз-контроль	Экономия денег
Антиблокировочная система	Безопасное торможение вместе с любимой семьей

Обязательно рассказывайте не о функциях, а о том, что в них полезного!

Чувствуете разницу?

Итак, первый шаг при оформлении продукции – это собраться вместе с командой и заказчиком и обсудить причины, по которым сотрудникам понравится с ней работать. Затем выберите три основные из них.

Этап 2. Создание слогана

Основа хорошего слогана – максимальное количество смысла в минимальном количестве слов. Не нужно описывать три перечисленные ниже компании, так как их слоганы говорят сами за себя.

- Acura – «За рулём или на ложе, а всё лучше помоложе»[5].
- FedEX – «Весь мир по расписанию»[6].
- Starbucks – «Рухни в прохладу»[7].

Чувствуете эмоции, заложенные в этих слоганах? Теперь расслабьтесь. Эти слоганы – одни из лучших, и ваш совсем не обязательно должен быть таким же профессиональным. Просто соберитесь с командой, выделите 10 или 15 минут на придумывание слогана и проявите свои творческие способности. Помните – ни один слоган не бывает чрезмерно банальным!

Этап 3. Разработка упаковки

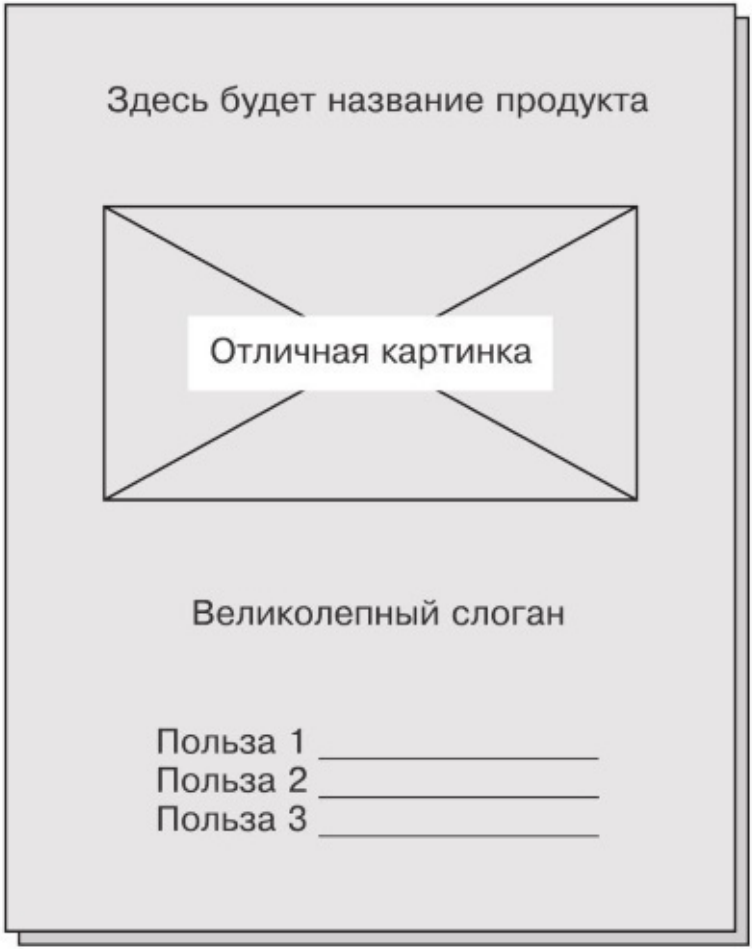
Отлично! Сформулировав три отличные причины купить ваш продукт и придумав для него запоминающийся слоган, вы готовы собрать все вместе.

---

[5] В оригинале: The true definition of luxury. Yours («Истинное воплощение роскоши. Вашей»). – *Примеч. пер.*

[6] В оригинале: Peace of mind («Спокойствие и уверенность»). – *Примеч. пер.*

[7] В оригинале: Rewarding everyday moments («Воздавая должное повседневности»). – *Примеч. пер.*



Выполняя эту часть работы, представьте себе, как покупа-

тель заходит в ближайший софтверный магазин и замечает на полке вашу программу. А когда клиент возьмет с полки коробку, она настолько ему понравится, что он сразу купит 10 штук – для себя и своих друзей.

Итак, переходим к оформлению упаковки!

Не пытайтесь сразу создать шедевр. Просто возьмите ватман, фломастеры, бумажные наклейки, иголочки и все, что еще может понадобиться. Озвучьте ваш слоган. Расскажите покупателям о пользе программы. Потратьте 15 минут на создание такой качественной упаковки, на которую вы только способны.

Прекрасно! Видите – не боги горшки обжигают. Надеюсь, вам понравился этот опыт (не каждый день приходится брать карандаши и рисовать выдающиеся изображения продукта). Такая практика отлично помогает сплотить команду и позволяет критически подойти к вопросу о том, что стоит за вашей программой.

Пришло время узнать, как дать клиенту представление о предполагаемом функционале нашего проекта.

4.4. Создание списка функций

БУДЕТ РЕАЛИЗОВАНО В ПРОГРАММЕ	НЕ БУДЕТ РЕАЛИЗОВАНО
Создание нового допуска. Обновление/чтение/удаление имеющихся допусков. Функция поиска. Базовая система отчетности. Печать	Взаимодействие с устаревшей системой блокировки проезда. Возможность работы офлайн
НЕ РЕШЕНО	
Интеграция с системой отслеживания логистики. Система считывания электронных карт допуска	

Знакомя клиента с ожидаемым функционалом проекта, важно рассказать не только о том, что вы собираетесь сделать, но и о том, чего вы делать *не планируете*.

Создавая список функций, которые вы не собираетесь реализовывать, вы ясно указываете, что входит в проект, а что – нет. Поступая так, вы не только позволяете клиенту составить верное представление о том, на что следует рассчитывать, но и гарантируете, что ваша команда сосредоточится при работе только на самом важном, абстрагировавшись от всего остального.

Конец ознакомительного фрагмента.