O'REILLY®

ПОЛЬЗОВАТЕЛЬСКИЕ ИСТОРИИ

Искусство гибкой разработки ПО

ПИТЕР®

Джефф Паттон

Бестселлеры O’Reilly (Питер)

Джефф Паттон

Пользовательские истории. Искусство гибкой разработки ПО

«Питер»
2014

УДК 004.42
ББК 32.973.2-018

Паттон Д.

Пользовательские истории. Искусство гибкой разработки ПО / Д. Паттон — «Питер», 2014 — (Бестселлеры O'Reilly (Питер))

ISBN 978-5-496-02931-5

Пользовательские истории – это метод описания требований к разрабатываемому продукту. В книге рассказано, как правильно использовать данную технику, чтобы сфокусироваться на поставленной задаче и пожеланиях клиента, а не распыляться на реализации второстепенных функций. Автор книги показывает, как данный подход не только ускоряет и систематизирует разработку, но и улучшает взаимопонимание в команде. В формате ios.epub сохранен издательский макет.

УДК 004.42
ББК 32.973.2-018

ISBN 978-5-496-02931-5

Содержание

Джефф Паттон
Пользовательские истории. Искусство гибкой разработки ПО

Посвящается Стейси, Грейс и Зоэ. Без вашей поддержки у меня ничего бы не вышло.

В память Люка Баррета, дорогого коллеги и учителя. Люк оказал огромное влияние на мою жизнь, а также на судьбы многих других людей.

Предисловие Мартина Фаулера

Одно из самых выгодных последствий популярности разработки программного обеспечения (ПО) по методологии Agile – распространение идеи разбиения больших, объемных требований на компактные фрагменты. Благодаря этим фрагментам – историям – отслеживать прогресс разработки проекта намного проще. Когда истории реализуют постепенно, каждый раз полностью интегрируя их в проект, всем очевидно, что проект понемногу растет. Рассматривая истории, которые приносят пользователям очевидную выгоду, разработчики могут планировать развитие проекта и определять, над чем нужно работать в следующую очередь. К тому же такая прозрачность подталкивает пользователей к активному участию в разработке – они больше не гадают месяцами и годами, чем занята команда разработки.

Тем не менее такое разбиение может иметь и негативные последствия. В частности, очень легко перестать понимать, в чем заключается общее предназначение ПО – что и как должна делать система. В итоге у вас в руках может оказаться множество кусочков, которые никак не складываются в единую картину. Или вы можете создать бессмысленную и бесполезную систему, так как утонули в деталях и забыли, что в действительности нужно пользователям.

Построение карт историй (story mapping) – это техника, позволяющая увидеть цельную картину, чего не удастся сделать с помощью простого набора историй.

Вот, собственно, и все – описание книги уместилось в одном предложении, но этого вполне достаточно, чтобы оценить преимущества метода. Обзор цельной картины облегчает взаимодействие с пользователями, позволяет избежать разработки ненужных функций, а также ориентирует на релевантный опыт использования. Когда я обсуждаю с коллегами по ThoughtWorks применяемый ими процесс разработки пользовательских историй, построение карт регулярно упоминается в качестве основной техники. Часто оказывается, что коллеги изучили эту технику как раз на семинарах Джеффа, поскольку именно он разработал ее и лучше всего может ей обучить. С помощью данной книги еще больше людей смогут узнать об этой технике непосредственно из первых уст.

Но эта книга не только для тех, у кого на бедже или в профиле написано что-нибудь вроде «бизнес-аналитик». Наверное, самым большим разочарованием для меня за 10 лет внедрения методологии Agile стало то, что множество программистов рассматривают истории как некие односторонние указания со стороны аналитиков. С самого начала предполагалось, что истории будут вызывать *обсуждение*. Если вы и в самом деле хотите получить эффективное ПО, которое может органично встроиться в человеческую деятельность, то тех, кто создает программы, необходимо рассматривать как живой источник идей о возможностях, ведь именно программисты лучше всех знают, что могут делать эти программы. Программисты должны хорошо понимать, что хотят получить их пользователи, и взаимодействовать с ними, создавая карты историй, где полностью учитываются пользовательские цели. Программист, умеющий составлять карты историй, может видеть пользовательскую среду куда более широко, чем тот, кто этого не умеет, и, следовательно, принимать участие в проектировании ПО, что улучшит качество работы.

Когда Кент Бек, впервые предложивший термин «история», воплотил свои идеи в разработке ПО, он назвал коммуникацию ключевым моментом эффективности команды. Истории – строительные блоки коммуникации между разработчиками и теми, кто использует результаты их труда. Карты историй организуют и структурируют эти строительные блоки и тем самым стимулируют процесс коммуникации, крайне важный для разработки ПО в целом.

Мартин Фаулер, 18 июня 2014 года

Предисловие Алана Купера

В научно-фантастическом романе Мэри Шелли «Франкенштейн» безумный доктор Франкенштейн создает чудовище из фрагментов тел мертвых людей, а затем оживляет его с помощью диковинной на тот момент силы электричества. Конечно, мы знаем, что на самом деле это невозможно. Вы не можете создать что-то живое, просто сшив вместе случайные части тел.

Тем не менее разработчики программного обеспечения все время пытаются сделать именно это. Они разрабатывают прекрасные новые функции для программ, одну за другой, а потом удивляются, почему лишь немногие пользователи восхищаются их продуктом. Ключ к загадке в том, что в качестве инструмента для проектирования и дизайна программисты используют свои методы разработки ПО, но эти средства совсем не взаимозаменяемы.

Более чем разумно *программировать* только одну функциональность ПО в каждый момент времени. Это идеальная стратегия, проверенная временем. Кроме того, многолетним опытом разработки было доказано, что использование такого подхода при проектировании цифровых продуктов, как одна функциональность в каждый момент времени, порождает монстров, подобных Франкенштейну, а не качественные программы.

Хотя процессы проектирования программного обеспечения и его непосредственной разработки тесно связаны, по своей сути они совершенно различны и, как правило, их выполняют разные люди с разным набором навыков. Если заставить программистов, подобно дизайнерам интерфейсов, проводить многие часы за наблюдением работы пользователей и выделением поведенческих паттернов, они просто на стену полезут. В то же время дизайнер, погрузившись в код и алгоритмы, почувствует себя выброшенным на необитаемый остров.

Но когда две составляющие одного процесса – проектирование и разработка – выполняются одновременно, работа идет искрометно, а у продукта есть все шансы родиться живым и дышащим. Именно командная работа вдыхает в монстра жизнь и заставляет людей полюбить его.

Хотя идея командной работы не является ни новой, ни особенно революционной, эффективно воплотить ее в жизнь нелегко. Особенности работы программистов – темп, ритм, привычный язык – сильно отличаются от методов, присущих дизайнерам.

Представители каждой из сторон могут быть решительными, способными, дисциплинированными, но у них есть одно общее слабое место: очень трудно описать дизайнерскую проблему в терминах программирования и так же нелегко сделать обратное. Две родственные дисциплины не имеют общего языка. Именно там, на стыке этих двух дисциплин, и работает Джефф Паттон.

Метод построения карт историй, созданный Джеффом, находят полезным разработчики, и точно так же его оценивают дизайнеры. Карты историй – Розеттский камень нашего цифрового века.

На самом деле гибкие методологии – не самая лучшая среда для дизайна приложений, несмотря на популярность противоположного мнения. Да, такая философия разработки дружественна дизайну, и это очень хорошо, но сама по себе она не поможет вам создать продукт, который понравится пользователям. В то же время мы столько раз видели, как отличный, хорошо документированный дизайн передается разработчикам – работающим по Agile или нет, – а они в процессе реализации ухитряются загубить самую его суть.

Метод построения карт историй Паттона перекидывает мост через эту пропасть. Основа дизайна взаимодействия – это выяснение мельчайших потребностей пользователей и верная их интерпретация. Дизайнерская история, являющаяся формальной версией пользовательской истории, остается неизменной на протяжении всего периода разработки.

Современный мир бизнеса доказал, что команде из 200–300 человек почти невозможно создать продукт, который понравится пользователям. В то же время сообществу стартапов известно, что команда из четырех-пяти человек *способна* создать небольшой продукт, который люди полюбят, но даже эти маленькие продукты со временем растут и теряют свой блеск. Большие программы используются большой аудиторией и решают сложные коммерчески успешные задачи. Ждать, что вы легко их изучите и в процессе работы станете получать удовольствие, сложно да и просто смешно.

Единственный путь создать большую программу, не похожую на ужасного Франкенштейна, – научиться объединять дисциплины проектирования и разработки программного обеспечения. Никто не умеет делать это лучше, чем Джефф Паттон.

Алан Купер, 17 июня 2014 года

Предисловие Марти Когана

Мне невероятно повезло – я имел возможность работать с представителями лучших в мире компаний и групп разработки разных технологий. Эти люди создают программы, которые вы любите и которыми пользуетесь каждый день. Люди, которые буквально меняют мир.

Кроме этого, мне часто приходилось помогать компаниям, у которых дела идут не так здорово. Это были стартапы, пытающиеся запустить хоть что-то работающее, прежде чем кончатся деньги. Компании покрупнее, выбивающиеся из сил в попытке воплотить в жизнь свои последние разработки. Команды, безуспешно пытающиеся повысить эффективность бизнеса. Лидеры, раздраженные тем, как много времени занимает переход от идеи к воплощению. Инженеры, конфликтующие с владельцами своих продуктов.

Из этого всего я вынес в первую очередь понимание того, насколько по-разному создают технологические продукты самые популярные компании на рынке и все остальные. И я не говорю сейчас о каких-то мелких различиях. Я имею в виду решительно все: подход руководителей к делегированию полномочий командам, способ взаимодействия команд, отношение организации к финансированию, комплектованию штата и выпуску продуктов, культуру, а также то, каким образом объединяют продукт, дизайн и технологии, чтобы разрабатывать самые эффективные решения для клиентов.

Эта книга называется «Пользовательские истории. Искусство гибкой разработки ПО», но очень скоро вы заметите, что она повествует о чем-то большем, чем такая простая, но мощная техника, как построение пользовательских карт историй. С помощью книги можно проникнуть в самую суть того, как команды сотрудничают, общаются и в конце концов приходят к созданию великолепных продуктов.

У многих из вас никогда не было возможности с близкого расстояния наблюдать за сильной командой в процессе работы над проектом. Возможно, опыт работы в компаниях, где вы трудились раньше или трудитесь сейчас, – все, что у вас есть. Поэтому я попытаюсь рассказать о том, насколько самые лучшие команды отличаются от всех остальных.

С благодарностью в адрес Бена Хоровица и его книги «Хороший менеджер продукта, плохой менеджер продукта» я приведу здесь лишь важнейшие различия между сильными и слабыми командами.

• У хороших команд есть четкое видение своего продукта, а каждый член команды страстно заинтересован в успехе. Плохие команды – просто наемники.

• Хорошие команды черпают идеи и вдохновение из системы ключевых показателей эффективности, наблюдения за клиентами, анализа полученных от клиентов сведений о результатах использования их продукта, а также из стремления постоянно применять новейшие технологии для эффективного решения проблем. Плохие команды получают требования из запросов заказчиков и отдела продаж.

• Хорошие команды понимают, кто их ключевые партнеры, им известны ограничения, которые вынужден учитывать бизнес клиентов, и поэтому они стараются находить решения, не только работающие для пользователей и заказчиков, но и учитывающие условия среды. Плохие команды просто выполняют требования партнеров.

• Хорошие команды компетентны во множестве техник, позволяющих быстро опробовать новые идеи для развития продукта и определить, какие из них следует воплощать в первую очередь. Плохие команды тратят часы на совещания, где пытаются составить списки приоритетов.

• В хороших командах обожают мозговые штурмы с участием лучших умов всей компании. Плохие команды ощетиниваются, если кто-то извне осмеливается внести какое-то предложение.

• В хороших командах инженеры, дизайнеры и менеджеры работают бок о бок, все время обмениваясь опытом и информацией о функционале, пользовательском опыте и технологических возможностях. В плохих командах эти специалисты разделены согласно своим обязанностям, а запросы одних к другим передаются через служебные записки и совещания, проводимые по расписанию.

• Хорошие команды постоянно пробуют новые идеи и вводят различные усовершенствования, но делают это осторожно, чтобы не навредить эффективности бизнеса. Плохие команды ждут разрешения что-то попробовать.

• У участников хороших команд непременно есть полный набор навыков для создания сильных продуктов, например, с хорошим дизайном взаимодействия. Плохие команды даже не знают, кто такие дизайнеры интерфейсов.

• В хороших командах заботятся о том, чтобы у инженеров ежедневно находилось время поработать с прототипом продукта для поиска идей по его улучшению. В плохих командах инженерам показывают прототипы на планировании спринта при оценке объема работы.

• Хорошие команды еженедельно напрямую общаются с конечными пользователями и заказчиками, чтобы лучше понять их и узнать их мнение о последних изменениях и идеях. Плохие команды считают, что достаточно собственного мнения.

• Хорошие команды знают, что не все их любимые идеи будут работать для заказчиков, но даже те, что будут, потребуют нескольких доработок, прежде чем приведут к получению желаемого результата. В плохих командах просто делают то, что записано в плане, довольствуясь датами совещаний и показателями качества.

• Хорошие команды понимают важность быстродействия и регулярных прогонов для успешного внедрения инноваций; им известно, что скорость обеспечивается правильной организацией работы, а вовсе не напряженным трудом. В плохих командах все жалуются на медленную работу, обвиняя в этом недостаточно усердно трудящихся коллег.

• После оценивания затрат на реализацию запроса хорошие команды берут на себя жесткие обязательства и стараются убедиться, что они трудятся над жизнеспособным решением, которое будет эффективно работать как для заказчиков, так и для бизнеса. Плохие команды жалуются, что им приходится работать на эффективность продаж.

• Хорошие команды выстраивают свою работу так, что могут немедленно оценить, как их продукт используется, и сделать выводы, базирующиеся на этих данных. Плохие команды считают аналитику чем-то, что хорошо бы иметь.

• Хорошие команды постепенно и непрерывно обновляют продукт, зная, что постоянный поток небольших обновлений означает стабильное и надежное решение для заказчиков. Плохие команды проводят ручное тестирование в конце огромной фазы разработки, а затем выкатывают сразу все обновления.

• Хорошие команды концентрируются на своей целевой аудитории. Плохие команды концентрируются на конкурентах.

• Хорошие команды устраивают вечеринку, когда достигают значительного улучшения ключевых показателей эффективности. Плохие команды празднуют финальный релиз чего-нибудь.

Я понимаю: вы, вероятно, хотите знать, что общего со всем этим имеют карты историй. Я уверен, вы удивитесь, поняв, в чем дело. По той же причине я преданный поклонник построения карт историй.

На моем пути встретилось не так уж много экспертов Agile, по моим меркам достаточно квалифицированных для того, чтобы оказать реальную помощь серьезной команде, разрабатывающей продукт, и поднять ее работу на тот уровень, в котором нуждается компания и которого она заслуживает. Джефф Паттон – один из них. Я наблюдал, как он в разгар разработки засучив рукава трудится вместе со всей командой. Я представлял его в компаниях, потому что

он эффективен. Команды обожают его, так как при всей своей компетентности он совершенно лишен высокомерия.

Время, когда менеджеры день-деньской собирали и документировали требования, дизайнеры концентрировались на косметических улучшениях, а инженеры тонули в коде, для самых лучших команд давно ушло в прошлое. Настало время стремиться к будущему и вам.

Марти Коган, 18 июня 2014 года

Об авторе

За 20 лет практической работы Джефф Паттон убедился, что не существует единственно правильного способа проектирования и разработки программного обеспечения, а вот неправильных путей существует великое множество.

Для помощи организациям в улучшении их работы Джефф использует более чем 15-летний опыт работы с широким спектром продуктов – от системы онлайн-заказа запасных частей для самолетов до электронных медицинских карт. В то время как многие процессы разработки концентрируются на скорости и продуктивности, Джефф уравновешивает эти факторы созданием продуктов, которые обеспечивают полезность для бизнеса и успех на рынке.

Джефф решил специализироваться на подходах Agile с тех пор, как работал в команде экстремального программирования в 2000 году. В частности, он специализируется на интеграции эффективного дизайна пользовательского взаимодействия и менеджмента продуктов в мощные инженерные методы.

В настоящее время Джефф работает как независимый консультант, тренер процессов Agile, тренер процессов дизайна продуктов и инструктор. Множество статей, эссе и презентаций, посвященных различным аспектам разработки продуктов Agile, можно найти на сайте agileproductdesign.com и в Crystal Clear Алистера Коберна. Джефф – основатель и модератор дискуссионной группы Yahoo! по теме юзабилити в Agile, колумнист в StickyMinds.com и IEEE Software, сертифицированный тренер Scrum, а также обладатель премии Agile Alliance's 2007 Gordon Pask за вклад в развитие Agile.

Вступление

Живи в этом, плавай в этом, смейся в этом, люби в этом,
А еще оно уберет подозрительные пятна с простыней,
Развлечет во время визита к родным
И превратит сэндвич в банкет.

Том Уэйтс. Проходите, не стесняйтесь

На самом деле эта книга должна была быть совсем небольшой… чем-то вроде памфлета.

Вообще-то я собирался всего лишь описать простую технику, которую назвал *построением карт историй*. Я вместе с другими создаю простые карты, чтобы облегчить совместную работу, а также представить себе ощущения, возникающие при использовании нашего продукта.

Карта историй помогает нам держать фокус на пользователях и их опыте, в результате чего взаимодействие улучшается и продукт становится несравнимо лучше.

Составлять карты до смешного просто. Работая вместе с другими, я озвучиваю историю работы с продуктом, записывая каждый шаг, предпринимаемый пользователем, на листочках-стикерах и наклеивая их слева направо. Затем мы возвращаемся к началу и обсуждаем каждый шаг в деталях, записывая подробности на листочках и наклеивая их сверху вниз под соответствующим шагом. В результате получается простая, напоминающая таблицу структура, излагающая историю слева направо и раскрывающая детали сверху вниз. Быстро и очень инте-

ресно. А эти детали образуют *бэклог (backlog)*[1] историй для наших проектов, разрабатываемых по Agile.

Сложно ли написать об этом книгу?

Оказывается, и в самых простых вещах могут скрываться сложности. Поэтому описание того, зачем вообще строить карту историй и что с ней делать после того, как она построена, а также различных способов ее использования заняло немало страниц. Многовато для простой методики, какой я ее считал.

Если вы используете процесс разработки, описанный в методологии Agile, то ваши бэклоги и так, наверное, заполнены пользовательскими историями (user story). Я думал: раз создание историй является настолько распространенной практикой, писать о них книгу будет напрасной тратой времени. Но, как оказалось, я ошибался. Через полтора десятилетия после того, как истории впервые были описаны Кентом Беком, они стали наиболее популярны, а также наименее правильно понимаемы и используемы, чем когда-либо. Это меня огорчает. А главное, это сводит на нет все выгоды, которые мы получаем от составления карт историй.

Поэтому в этой книге я хотел бы скорректировать как можно больше недоразумений, связанных с использованием историй в разработке программного обеспечения по методологиям Agile и Lean. Вот почему, говоря словами Тома Уэйтса, я «превращаю этот сэндвич в банкет».

[1] Бэклог – набор функциональностей, которые планируется внедрить в проект. – *Примеч. пер.*

Почему я?

Я люблю создавать. Когда я разрабатываю какую-нибудь функциональность для программного обеспечения, а люди с удовольствием ею пользуются, это очень радует и мотивирует меня. И я не слишком люблю методологии. Я бы сказал, мне надо разобраться в принципе какой-то методики или практики, чтобы как следует ею овладеть. Только сейчас, имея более чем 20-летний опыт разработки программного обеспечения, я начинаю понимать, как учить других тому, что умею сам. Я также понимаю, что то, чему я учу, постоянно меняется. На этой неделе я что-то изучил, а на следующей оно уже изменилось. Способы объяснить это другим людям меняются почти так же часто. Все это многие годы удерживало меня от написания книги.

Но время наконец пришло.

Несомненно, истории и построение карт – вещь прекрасная. Множество людей извлекли из них пользу. Использование историй благотворно повлияло как на рабочий процесс, так и на продукт, который создавался. Но в то время, как одни люди замечали улучшения, другие, которых было больше, мучились во время работы с историями больше, чем когда-либо прежде. Вот это мне и хотелось бы прекратить.

Изменить ситуацию я надеюсь с помощью этой книги. Если мне удастся добиться хотя бы небольших улучшений, я буду считать это успехом.

Если вы используете истории и страдаете, эта книга – для вас

Уже довольно много организаций внедрило у себя методологии Agile и Lean, поэтому, вполне возможно, вы уже успели угодить в одну из ловушек, возникающих из-за неверного понимания концепции историй. Вот некоторые из них.

• Поскольку истории позволяют вам сконцентрироваться на создании небольших фрагментов ПО, легко *перестать видеть цельную картину*. В результате получается типичный продукт-франкенштейн, каждому пользователю которого очевидно, что он состоит из разрозненных, не связанных друг с другом частей.

• Когда вы работаете над продуктом значительных размеров, создание маленьких частичек одна за другой *заставляет людей задумываться, когда же вы наконец закончите и что же получится в результате*. Как будто вы строитель.

• Поскольку главное в концепции историй – это обсуждение, *люди часто забывают вести записи*. В результате предмет обсуждения и достигнутые соглашения забываются.

• Поскольку в хороших историях предполагается наличие критериев приемки, мы концентрируемся на определении этих критериев. Но этот процесс и описание создаваемого продукта – не одно и то же. В результате *команда не может закончить запланированную работу в запланированные сроки*.

• Поскольку хорошие истории должны быть написаны с позиции пользователя, но существует множество аспектов, которых пользователь просто не видит, члены команды утверждают: «У этого продукта нет пользователей, так что здесь пользовательские истории не подходят».

Если вы уже угодили в одну из этих ловушек, я постараюсь прояснить все вызвавшие их недоразумения. Вы узнаете, как оценить полную картину, продуктивно обсуждать цели и задачи пользователей и создавать хорошее ПО.

Для кого еще эта книга

Для вас, конечно. Особенно если вы ее уже купили. Я вот считаю, что вы сделали умную инвестицию. Если же просто взяли у кого-то книгу почитать, лучше закажите свой экземпляр, а эту верните, как только получите собственную.

Во всяком случае, чтение книги будет особенно полезным для специалистов следующих областей.

Продукт-менеджеры и UX-специалисты в коммерческих компаниях, производящих продукты, должны прочесть эту книгу, чтобы перекинуть мостик от мира пользовательского опыта и работы продукта к тактическим планам и элементам бэклога. Если вы испытываете трудности, пытаясь перейти от представления о продукте к отдельным деталям, которые должна создать ваша команда, истории вам помогут. Если вам сложно заставить других людей поставить себя на место пользователей, карта историй вам поможет. Если вы никак не можете увязать вместе хороший дизайн взаимодействия и практическое проектирование продукта, вам поможет эта книга. И если пытаетесь провести эксперимент в стиле стартапа Lean, она тоже будет вам полезна.

Представители заказчиков, бизнес-аналитики, а также продукт-менеджеры в организациях, занятых в сфере информационных технологий, должны прочесть эту книгу, чтобы возвести мосты между пользователями, разработчиками и другими заинтересованными сторонами. Если вы тратите множество усилий, чтобы все заинтересованные лица в вашей компании пришли наконец к какому-либо соглашению, карты историй вам помогут. А если разработчики затрудняются, пытаясь нарисовать цельную картину, истории будут полезны и здесь.

Тренеры процессов Agile и Lean, если они хотят помогать командам и отдельным людям действовать эффективнее, должны прочесть эту книгу. Кроме того, подумайте только, насколько неверное представление об историях сформировалось у сотрудников вашей организации! Применяйте истории, простые упражнения и практики, описанные в этой книге, чтобы помочь вашим командам развиваться.

И наконец, все остальные. При использовании процессов Agile мы чаще всего ожидаем плодотворной работы с историями от людей, исполняющих обязанности бизнес-аналитиков или представителей заказчиков, но по-настоящему эффективной работа станет, если основы будут известны *всем*. Если люди не понимают самых простых вещей, вы часто будете слышать жалобы, что истории плохо расписаны, или слишком длинные, или недостаточно детализированы. Эта книга поможет, но не так, как вы ожидаете. Вы вместе с другими читателями узнаете, что создание историй – это не способ лучше писать требования, а путь к более продуктивным и организованным обсуждениям. Эта книга поможет вам сформулировать, какие виды обсуждений необходимы, чтобы в любой момент у вас имелась нужная информация.

Надеюсь, вы отнесли себя к одной или нескольким из описанных здесь групп. Если нет, подарите эту книгу кому-нибудь, кто им соответствует.

А если это все-таки вы, давайте начнем.

Используемые соглашения

Как я полагаю, это не первая книга о разработке программного обеспечения, которую вы читаете, поэтому ничто не должно вас особенно удивить.

Подзаголовки внутри каждой главы будут подсказывать вам направление темы

Обращайте на них внимание, чтобы найти что-либо нужное или пропустить материал, неинтересный вам в данный момент.

> *Ключевые моменты оформлены примерно так. Представляйте, что это нужно прочитать чуть более громко, чем остальной текст.*

Если вы читаете по диагонали, двигайтесь от одной ключевой точки к другой. Если вас что-то заинтересовало или просто вы не находите сказанное банальным, прочтите предыдущий и последующий текст. Таким образом вы разберетесь в деталях.

> Врезки используются для описания:
>
> • интересных, но не критически важных аспектов. Возможно, они развлекут вас, по крайней мере я на это надеюсь;
>
> • конкретных упражнений. Вы сможете использовать их, чтобы начать практиковаться в чем-либо специфическом;
>
> • историй и примеров, предоставленных другими. Возможно, вы почерпнете из них хорошие идеи и сможете применить их в своей компании.

Эта книга разбита на разделы. Вы можете читать по разделу за раз или обращаться к ним, чтобы найти какие-то идеи для решения конкретных задач, занимающих вас в данный момент.

Как устроена эта книга

Однажды я купил замечательный лазерный принтер. Первое, что я увидел, открыв коробку, был бумажный буклет, на котором красовалась надпись большими яркими буквами: «СНАЧАЛА ПРОЧТИТЕ ЭТО». Я задумался, стоит ли так поступать на самом деле, ведь, как правило, я пренебрегаю инструкциями. Но позже очень радовался, что послушался этого предупреждения, потому что, как оказалось, в разных местах внутри принтера было закреплено много пластиковых деталей, чтобы уберечь его от повреждений при транспортировке, и, если бы я включил принтер, не убрав их, скорее всего, он был бы безнадежно испорчен.

Эта история, наверное, кажется здесь неуместной, но это не так.

Книга тоже содержит главу «Сначала прочтите это», где описаны две критически важные концепции, а также приводятся термины, которые я буду использовать в тексте. Мне хотелось бы, чтобы вы ознакомились с этим материалом, прежде чем приступите к дальнейшему чтению. Если вы начнете работать с картами историй прежде, чем хорошенько усвоите эту главу, я не могу гарантировать вашу безопасность.

Построение карт историй с высоты птичьего полета

Главы 1–4 дадут вам общее представление о построении карт историй. Если вы уже используете их и имеете некоторый опыт, эта часть книги обеспечит достаточно материала, чтобы немедленно приступить к делу.

Глава 5 предоставит вам отличную возможность попрактиковаться в ключевых концептах, используемых для составления наилучшей карты историй. Попробуйте проделать это в своем офисе с группой коллег – в результате каждый участник получит полезный опыт. Я обещаю: созданные вами карты со временем будут становиться все лучше и лучше.

Интуитивное понимание пользовательских историй

В главах 6–12 рассказано многое о пользовательских историях: как они работают в реальности, как организовать их использование в проектах Agile или Lean наилучшим образом. В дополнение к картам историй там приведены несколько маленьких примеров, которые могут оказаться полезными в ежедневной практике разработки. Даже если вы ветеран Agile, обещаю, вы узнаете об историях что-то новое для себя. А если вы в историях новичок, то узнаете достаточно, чтобы поразить самого заносчивого Agile-всезнайку в офисе.

Лучшие бэклоги

Главы 13–15 раскроют перед вами подробности жизненного цикла историй. Мы обсудим конкретные практические приемы, которые помогут использовать истории и карты историй. Постепенно мы пройдем от открывающихся перспектив к составлению бэклога, заполненного историями, описывающими жизнеспособный продукт. Вы узнаете, как построение карт историй и другие приемы могут помочь вам на каждом этапе работы.

Лучшая разработка

Главы 16–18 шаг за шагом посвятят вас в тонкости тактики использования историй. Вы научитесь доводить истории до конца, не упускать их из виду в процессе разработки, доби-

ваться их точного исполнения, а также извлекать опыт из каждой истории, трансформированной вами в рабочее ПО.

Я обнаружил, что последние несколько глав в большинстве книг по разработке ПО – просто бесполезный перевод бумаги. Обычно их можно безболезненно пропустить. К сожалению, в моей книге я ничего такого не написал и вам придется прочесть ее целиком. Могу утешить вас лишь тем, что в каждой главе вы найдете какие-то полезные приемы и уроки, которые сможете немедленно применить на практике.

Перейдем к делу.

Сначала прочтите это

В этой книге нет введения.

Да, вы все прочитали правильно. И сейчас, наверное, задаетесь вопросом: «Почему же в книге Джеффа нет введения? Может быть, он забыл его написать? Не пора ли ему на покой? Или введение съела его собака?»

Нет, я не забыл написать введение. И мне не пора на покой (по крайней мере я туда не собираюсь). И моя собака не съедала введения, хотя насчет морской свинки своей дочери я не был бы так уверен. Это просто потому, что за долгое время я убедился: авторы тратят кучу времени, пытаясь убедить меня прочесть их книгу, и большая часть этого времени приходится на введение. Самое интересное во всех книгах начинается примерно с третьей главы. Кроме того, я уверен, что не я один всегда пропускаю введение.

Так что эта книга начинается прямо здесь.

И вам ни в коем случае нельзя пропускать эту часть, так как на самом деле она самая важная. Я буду доволен, если вы вынесете из книги всего две мысли. Вот эти две мысли.

- Цель работы с историями не написание идеальных историй.
- Цель разработки продуктов не создание продуктов.

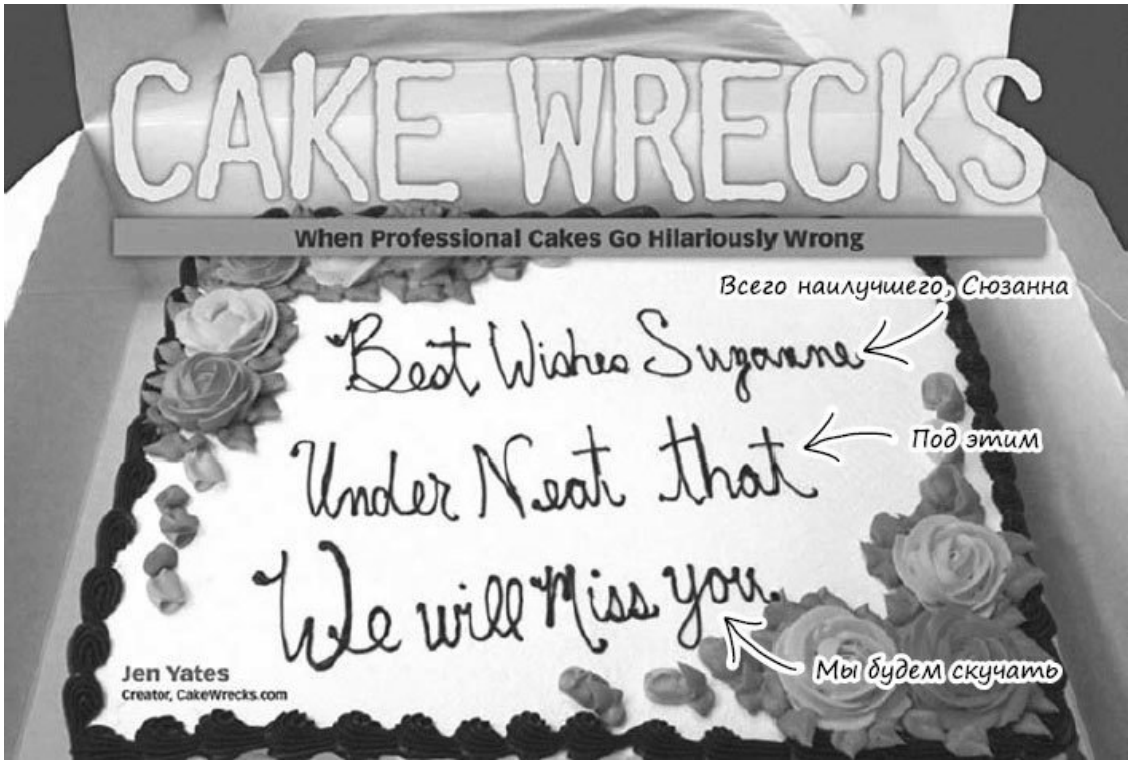
Сейчас я все объясню.

Игра в испорченный телефон

Вы наверняка помните, как в детстве играли в дурацкую игру «Испорченный телефон». Вы говорили кому-то шепотом слово или фразу, тот озвучивал это следующему, и так повторялось с каждым играющим, а затем последний участник провозглашал совершенно искаженное сообщение, и все дружно смеялись. Сегодня мы часто играем в эту игру всей семьей за обедом. Кстати, родителям на заметку: это отличный способ занять детей, скучающих, когда взрослые беседуют.

В мире взрослых мы также продолжаем играть в эту игру, разве что больше не шепчем. Мы пишем длиннющие документы и проводим торжественные презентации, чтобы дать поручение кому-то другому, а затем получить от него совершенно не то, чего ожидали. А этот человек использует эти документы, чтобы написать еще больше документов и передать их еще большему количеству людей. Только вот, в отличие от детской игры, в конце концов нам становится не до смеха.

Когда люди читают письменные инструкции, они истолковывают их совершенно по-разному. Если вам сложно в это поверить (в конце концов, все это тоже написано!), читайте дальше – вот несколько примеров того, как инструкции действуют абсолютно неверно.



Перед вами обложка книги Джен Ятис «Торты, которым не повезло», опубликованной в издательстве Эндрю Мак-Милла (спасибо Джен и Джону Ятис за предоставление иллюстрации). Книга получилась по мотивам ее очень забавного сайта cakewrecks.com (только не ходите туда, если у вас нет по меньшей мере часа свободного времени). На сайте собрана коллекция по-идиотски украшенных тортов, логика их создателей не поддается объяснению, хотя Джен и предпринимает попытки сделать это. Так, одна из самых часто встречающихся и в книге, и на сайте тем – неверно понятые требования. Джен, конечно, не называет их *требованиями*, ведь это слишком формальный термин, вместо этого она употребляет слово *«записи»*, так как исполнитель слушает и записывает, понимая буквально то, что слышит. Глядя на эти фото, я

могу вообразить сотрудника кондитерской, который слушает заказчика и записывает его пожелания, а потом передает их кому-то, кто будет украшать торт.

Заказчик: «Здравствуйте, я хотел бы заказать торт».

Работник: «Конечно, что бы вы хотели на нем написать?»

З.: «Вы могли бы написать “Всего хорошего, Алиса” фиолетовым цветом?»

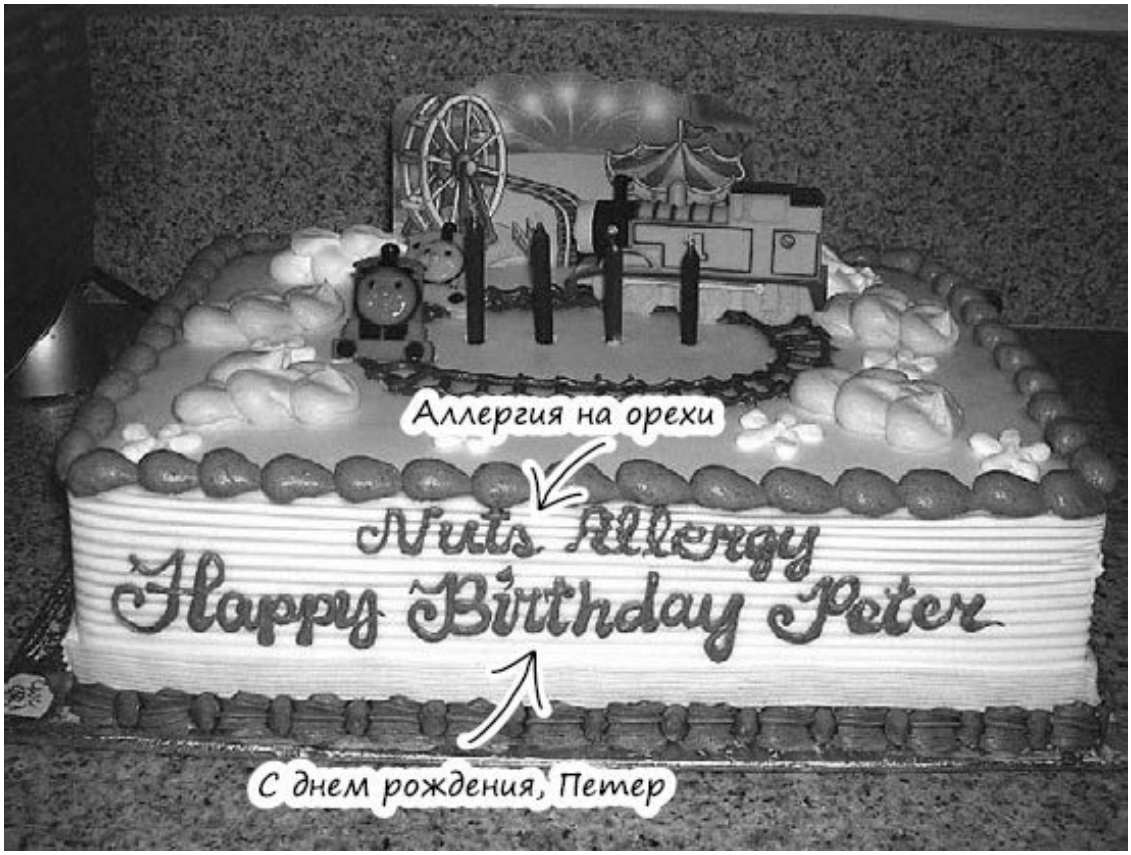
Р.: «Конечно».

З.: «А вокруг надписи пусть будут звезды».

Р.: «Без проблем. Я записал ваши пожелания и прямо сейчас передам их кондитеру-декоратору. Торт будет готов к завтрашнему утру».

Вот что получилось в результате.

Вот еще один пример. В разработке программного обеспечения такие вещи мы называем *нефункциональными требованиями*.



Все это очень весело, и можно посмеяться над 20 долларами, пропавшими зря. Но часто понесенные потери бывают куда более серьезными.

Может быть, вы слышали о неудачной попытке отправить в 1999 году на Марс орбитальный климатический зонд, в результате аварии которого NASA понесло убытки в размере 125 млн долларов[2]? Если не слышали, вот суть случившегося. Если когда-нибудь какой-либо проект по самые уши тонул в бумажной документации, это был, несомненно, проект NASA. Но несмотря на огромное количество требований и других документов, зонд вышел из строя по той простой причине, что NASA пользовалось метрической системой измерений, а инженеры партнерской компании Lockheed Martin, которые разрабатывали навигационные команды для двигателей аппарата, – британской. В результате никто не знает, где сейчас находится зонд, и некоторые надеются, что он нашел свое место на солнечной орбите где-то за Марсом.

Какая ирония – мы вкладываем огромные усилия в написание документов, чтобы общаться более ясно и избегать недоразумений, а в итоге получаем прямо противоположный результат.

Общие документы не дают единого понимания.

Притормозите на минутку и запишите это. Запишите на стикере и положите себе в карман. Представьте, что эти слова вытатуированы где-то на вашем теле и вы можете их видеть, когда утром приступаете к работе. Когда вы прочитаете их снова, вспомните истории, которые я вам сейчас рассказываю.

Единое (общее) понимание – это когда мы оба хорошо понимаем, что и почему представляет себе каждый. Очевидно, что между кондитерами и людьми, дававшими им инструкции, такого понимания не было. И в NASA какой-то важный начальник не обеспечил одинакового

[2] Подробности этой истории многократно освещались в прессе, вот одна из статей: http://edition.cnn.com/TECH/space/9909/30/mars.metric.02/. – *Примеч. пер*

понимания того, как работает система управления, всеми участниками процесса. Уверен, если вы трудитесь в сфере разработки ПО, то быстро вспомните аналогичную ситуацию: два человека были уверены, что пришли к соглашению относительно функции, которую хотели добавить в проект, но оказалось, что они представляли себе совершенно разные вещи.

Единое понимание – это невероятно просто

Мой бывший коллега Люк Баррет первым показал мне этот комикс, чтобы описать проблему. Я спросил, где он его видел, но он так и не вспомнил (кто-то, возможно, не получает авторских отчислений). Годами я наблюдал, как Люк демонстрирует эти четыре слайда в презентации PowerPoint, и всегда воспринимал их как нечто забавное, но довольно банальное. Видимо, я туговато соображаю: лишь много лет спустя осознал, что этот комикс иллюстрирует, наверное, самую важную особенность работы с историями при разработке ПО.

Суть в том, что если у меня есть какая-то идея, я ее записываю на бумаге, а *вы* затем это читаете, то вполне можете представить что-то совершенно иное, чем я. При этом можно спросить каждого: «Вы согласны с содержанием этого документа?», и все дружно ответят: «Да! Да, мы согласны!» Но если мы соберемся вместе и начнем обсуждение, вы можете описать мне, что думаете, а я смогу задать вопросы. Беседа пойдет живей, если мы можем описать наши мысли, рисуя картинки либо фиксируя идеи на стикерах или карточках. Если у каждого из нас будет возможность объяснить свои мысли с помощью слов и картинок, мы придем к общему мнению. Однако здесь мы осознаем, что все понимают всё по-разному, и это очень неприятно. Но по крайней мере теперь проблема известна.

Это не значит, что один из нас прав, а другой – нет, напротив, мы оба видим разные и важные аспекты. Комбинируя и уточняя наши различающиеся представления, мы в конце концов придем к общему мнению, включающему все лучшие идеи. Вот почему вынесение идей вовне так важно. Мы можем перерисовывать эскизы или передвигать с места на место стикеры – таким образом мы в буквальном смысле держим в руках свои идеи, и это замечательно. В этот момент мы формируем общее мнение, а это очень тяжело сделать, оперируя только словами.

По окончании обсуждения мы по-прежнему можем упоминать те же самые функции или предлагать улучшения, но сейчас определенно говорим об одних и тех же вещах. Мы синхронизированы и точно знаем, что движемся вместе в одном направлении. Вот качественный уровень, к которому мы стремимся. И, к сожалению, неосязаемый. Вы не можете увидеть или потрогать общее мнение, но можете его почувствовать.

Перестаньте пытаться написать идеальную документацию

Огромное количество людей до сих пор верит, что есть некий идеальный способ составлять документы. Поэтому они думают, что, когда люди читают документ и у них возникают некие разногласия, это происходит лишь из-за небрежности автора или читателя. На самом деле ни то ни другое не верно.

Правильный ответ: просто не занимайтесь документами.

Перестаньте пытаться составлять идеальные документы.

Возьмите и напишите что-нибудь, неважно как. Затем проведите эффективное обсуждение с использованием и слов, и картинок, чтобы прийти к общему мнению.

> *Истинная цель использования историй – достижение единого понимания.*

Если вы применяете в разработке истории, но не обсуждаете их с командой с помощью слов и рисунков, то вы неправильно используете истории.

Если при чтении этой книги вы поставили перед собой цель научиться лучше писать истории, то это плохая цель.

Хорошие документы похожи на фотографии из отпуска

Если я покажу вам одну из фотографий, сделанных во время отпуска, вы увидите моих дочерей на пляже и вежливо скажете что-то вроде: «Очень красиво». Но когда я сам смотрю на это фото, то вспоминаю определенный пляж на Гавайях, куда нам пришлось больше часа ехать на машине по ухабистой грунтовой дороге, а затем еще полчаса идти по лавовым полям. Я помню, как дети ныли и жаловались, уверяя, что ничего не может быть хуже, и я был с ними вполне согласен. Но, придя наконец на место, мы провели великолепный день на потрясающем, почти безлюдном пляже, ради чего мы и решились на эту поездку. Кульминацией этого чудесного дня стали черепахи, пришедшие погреться на песке у воды.

Конечно, когда вы смотрите на фотографию, вы не можете знать всего этого, так как не были там. А я помню, потому что я там был.

Хорошо это или плохо, но именно так работают документы.

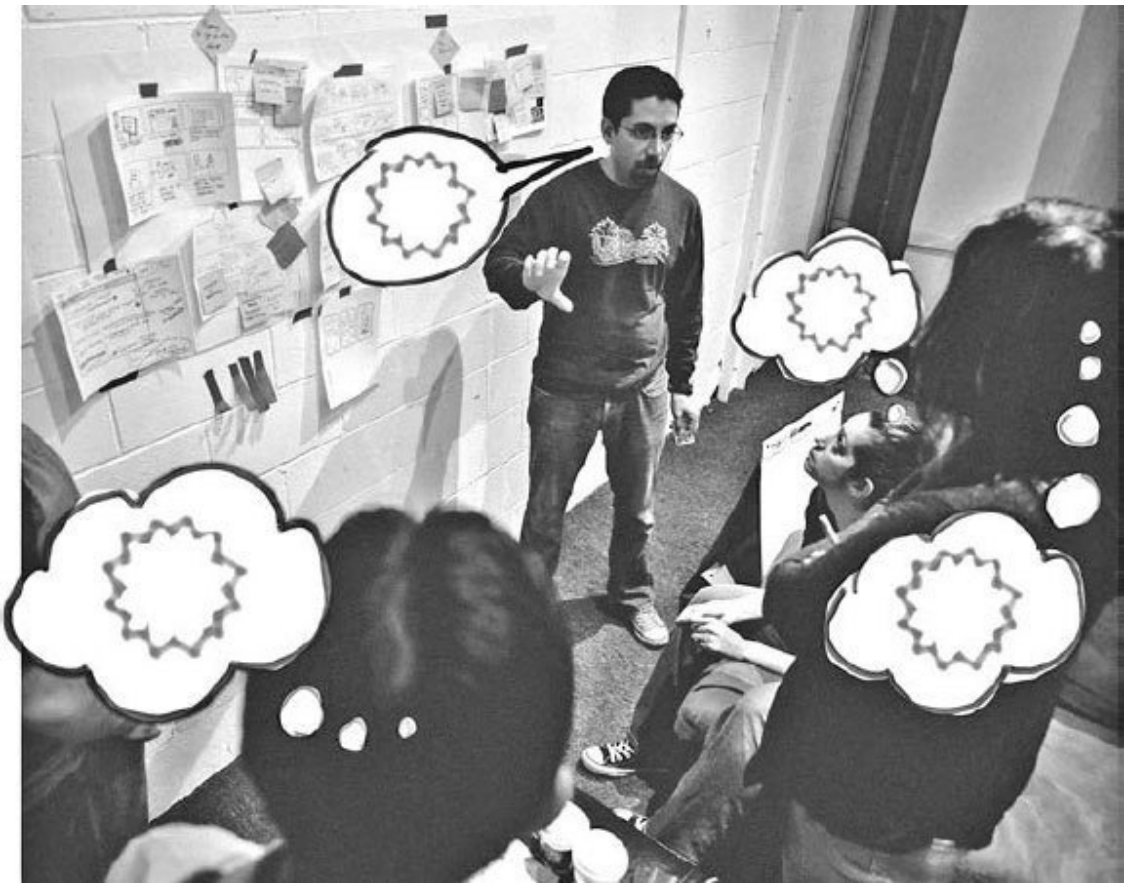
Если вы принимаете участие во множестве обсуждений создаваемого продукта, а затем документируете полученные результаты, то обязательно должны показать документ кому-то еще, кто тоже при этом присутствовал. Вы оба можете согласиться, что он хорош. Но помните: единое понимание заключается в деталях, которые в нем не отражены. Другой читатель, не присутствовавший при обсуждении, не извлечет из документа всего, что знаете вы. Даже если он говорит, что все понятно, не верьте этому. Найдите время, чтобы на основе документа рассказать историю точно так же, как я рассказал о своем отпуске, используя фотографию.

Документируйте, чтобы активизировать воспоминания

Я пару раз слышал шутку: «Мы перестали писать документацию, и поэтому у нас теперь Agile». Это называется: кто знает – тот поймет, потому что на самом деле процесс, управляемый историями, требует для работы множества документов. Но зачастую эти документы совсем не похожи на традиционные задокументированные требования.

Применять Agile – значит много обсуждать, рисовать эскизы, записывать, возиться с карточками или стикерами. Приносить на обсуждение документы, а затем уносить их, испещренные пометками на полях и исчерканные маркерами. Воздух наполнен энергией и движением. Если вы сидите в конференц-зале и кто-то один сводит все сказанное в систему управления историями, определенно здесь что-то не так.

Когда вы описываете историю, почти все может быть использовано как инструмент коммуникации. А поскольку мы обсуждаем истории, пишем массу заметок и рисуем кипы рисунков, надо их где-то хранить. Мы складываем их, чтобы позднее просмотреть, фотографируем, включаем в разные документы.



Но помните: самое важное и нигде не записанное – это то, что мы вспомним при прочтении. Вот он, фактор фотографии из отпуска.

Обсуждайте, рисуйте эскизы, записывайте, наклеивайте стикеры, а затем фотографируйте полученные результаты. Лучше даже запишите краткое видео команды, обсуждающей записанное на доске. Вы вспомните массу очень важных деталей, чего никогда нельзя добиться документированием.

Чтобы облегчить вспоминание, фотографируйте, а также записывайте короткие видео по результатам обсуждений.

Обсуждайте то, что действительно нужно

Многие люди уверены, что их работа – формирование и сбор требований. Но это не так.

На самом деле ваша работа – изменить мир.

Я сказал это, чтобы привлечь ваше внимание, и понимаю, что это звучит слишком пафосно. Эта фраза и в самом деле обычно ассоциируется с миром во всем мире, ликвидацией нищеты и тому подобными вещами вроде попыток политиков договориться друг с другом. Но вообще-то я серьезно. Любая отличная идея, которую вы реализуете как решение для продукта, в небольшой, а может, в значительной степени меняет мир людей, использующих этот продукт. Если это не так, вы потерпели неудачу.

До и после

Есть маленькая модель изменения мира, которую лично я использую и всегда держу в голове, вы тоже должны помнить о ней на протяжении всего обсуждения историй и выстраивания одинакового понимания.

Вот как я изображаю эту модель.

Модель начинается с исследования того, каков мир в настоящий момент, до начала работы. Когда вы смотрите на мир в состоянии «до», то предполагаете обнаружить людей недовольных, раздраженных, растерянных или злых. Однако мир очень велик, поэтому мы концентрируемся в основном на тех, кто работает с нашим ПО, а также на потенциальных его пользователях. Когда вы понаблюдаете, чем они занимаются, какие инструменты используют и как именно берутся за дело, скорее всего, вам в голову придет несколько идей, например таких.

- Какие новые продукты вы можете создать.
- Какие функции добавить в существующий продукт.
- Как улучшить создаваемые продукты.

В какой-то момент вам нужно будет обсудить свои идеи с другими людьми, после чего можете начать проектировать дизайн и писать *спецификации*. Если вы планируете передать эту работу кому-то другому, то и в самом деле можете употреблять слово *«требования»*. Но очень важно помнить, что требования – просто другое название возникших у нас идей о том, как помочь людям.

Имея эти требования, мы движемся по некоему процессу и в результате выпускаем продукт – какое-то программное обеспечение, которое выходит в свет – в мир в состоянии *«после»*. Мы с вами очень надеемся, что люди, которые изначально были недовольны, раздражены, растеряны или злы, после этого станут счастливыми. И они счастливы не потому, что им принесли красивую коробку с бантиком, ведь программное обеспечение в наши дни не доставляется в коробках. И не потому, что они прочитали протоколы изменений или скачали мобильное приложение в свой телефон. Они счастливы, потому что, используя программу, или сайт, или мобильное приложение, или что-то еще, созданное вами, замечают изменения к лучшему.

Конечно, вам не удастся угодить всем сразу. Сказать вам об этом должна была мама. Одни люди будут радоваться любым изменениям больше остальных, а другие так и останутся

недовольными, каким бы тяжелым ни был ваш труд и как бы вы ни старались найти самое лучшее решение.

Суть не в программах

Все, что находится между идеей и выпуском продукта в свет, называется *объемом работы*. Это то, что мы создаем. Люди, занятые разработкой программного обеспечения по Agile, обычно измеряют *скорость* работы и стараются увеличить ее. Те, кто создает ПО, разумеется, беспокоятся о стоимости готового продукта, в том числе о временных затратах на его создание, планируемых и реальных.

Но в таком случае объем работы не является основным, ведь мы заботимся не о количестве работы. Нас интересует то, что получается в итоге, – *реальный результат*. Реальный результат – то, что получается в результате работы (отсюда и название), и его нелегко оценить, поскольку невозможно измерить результат, не доведя работу до конца. Кроме того, реальный результат нельзя измерить количеством добавленных функций или новыми возможностями, предъявленными пользователям. По сути, мы фиксируем, что именно люди делают иначе для достижения своей цели теперь, в результате сделанных вами изменений, и, самое главное, стала ли их жизнь несколько лучше[3].

Ну вот вы и изменили мир.

Вы добавили в него что-то изменяющее способ достижения людьми своих целей, и когда они это используют, мир для них меняется.

Если помните, наша цель – не создать новый продукт или функцию. Если вы обсуждаете эту функцию, то стараетесь понять, для кого она предназначена, как эти люди справляются со своими задачами сейчас и что можно поменять для них в будущем. Ожидание позитивных изменений мотивирует людей к использованию вашего продукта.

> *Плодотворное обсуждение историй включает в себя вопросы «Для кого?» и «Почему?», а не только «Что?».*

[3] В точной терминологии, а также разнице между объемом работы и конечным результатом я разобрался, прослушав речь Роберта Фабриканта Behavior is Our Medium. До этого я, как и многие другие, затруднялся внятно изложить то, что хорошо понимал. К счастью, Роберту удалось внести ясность. – *Примеч. авт.*

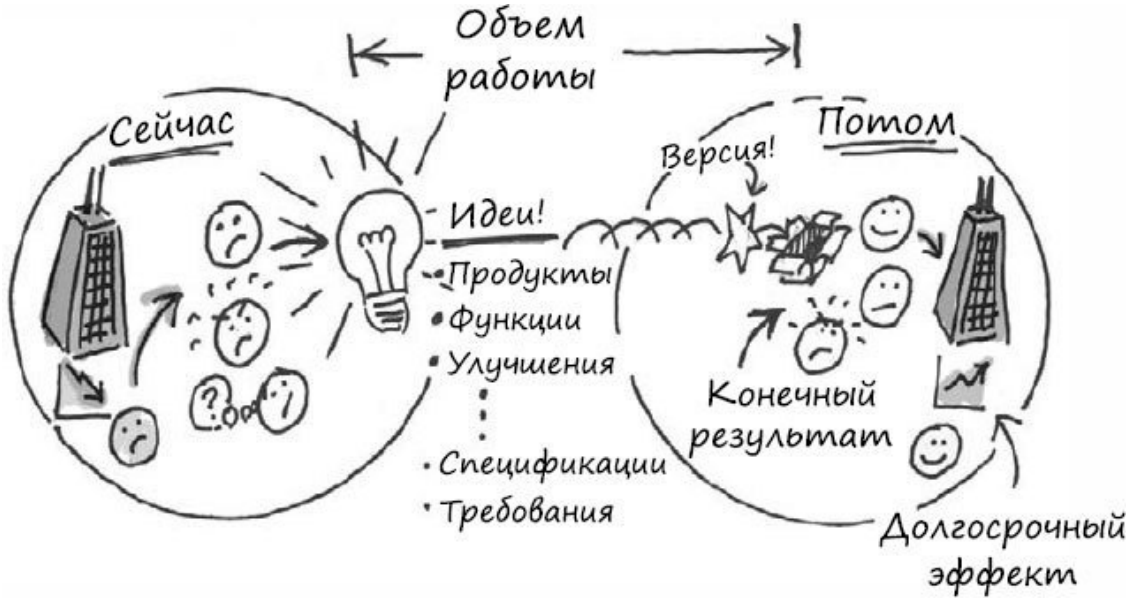
Ладно, не только о людях

Я думаю о людях не меньше, чем все остальные, но, понятное дело, мы говорим не только о том, чтобы сделать их счастливыми. Если вы работаете в компании, которая платит вам и другим работникам зарплату, вам приходится концентрироваться на том, что конкретно больше всего помогает вашей организации зарабатывать больше денег, защищать свой рынок или захватывать новые, работать более эффективно. Ведь если в вашей компании дела идут не очень, то у вас нет средств (или работы), чтобы кому-нибудь помочь.

Поэтому модель придется немного переработать. На самом деле начинать нужно с обзора вашей организации изнутри. Там вы найдете еще больше людей, недовольных жизнью, чаще всего потому, что их бизнес работает не так хорошо, как они надеялись. Чтобы улучшить положение вещей, эти люди чаще всего предлагают сфокусироваться на конкретных пользователях или заказчиках с целью создать или усовершенствовать нужные им продукты. В итоге, как видите, все сводится к людям, потому что

> *ваша компания не получит того, чего хочет, пока заказчики или пользователи не получат того, чего хотят они.*

Развитием процесса становится выбор людей, на которых мы будем фокусироваться, проблем, которые нужно решить, а затем идей, которые можно внедрить в работающее ПО. После этого – если заказчики покупают, пользователи используют и все они довольны, – бизнес, который спонсировал данную разработку, начинает получать выгоду. Это может выразиться в возрастании прибыли, снижении издержек, повышении лояльности заказчиков, увеличении доли рынка. Таким образом, многие сотрудники вашей компании получают удовлетворение. Это и вас должно порадовать, ведь вы только что помогли своей компании получить выгоду, одновременно принеся пользу реальным людям. Это стратегия «вин-вин», или стратегия обоюдной выгоды.



Вот какими могут быть последствия получения хорошего реального результата – я называю их *долгосрочным эффектом*. Реальные результаты вы чаще всего видите сразу после выпуска продукта, но долгосрочный эффект этой работы проявляется позже.

Программируйте меньше

Да, это неприятная правда в мире программного обеспечения и, как я подозреваю, во множестве других сфер. Но я хорошо знаю эту кухню. И мне точно известно, что

> *у нас никогда не будет достаточно времени или ресурсов, чтобы разработать все, что нужно, – никогда!*

Одним из самых больших недоразумений в разработке ПО является то, что мы пытаемся выдать как можно больший результат за как можно меньший промежуток времени. Да, если бы у нас было слишком много работы, то увеличение скорости, конечно, помогло бы, верно? Но если вы трезво посмотрите на ситуацию, то поймете, что наша работа заключается не в том, чтобы сделать больше, – она в том, чтобы сделать меньше.

Конец ознакомительного фрагмента.