Сделано

Скотт Беркун

Проектный
менеджмент
на практике

O'REILLY®

Скотт Беркун

Сделано

«Манн, Иванов и Фербер (МИФ)»

2008

УДК 658.5:004
ББК 65.291.21с5165.291.21с51

Беркун С.

Сделано / С. Беркун — «Манн, Иванов и Фербер (МИФ)», 2008

Скотт Беркун долгое время работал в Microsoft и делится с читателями своим опытом управления проектами. Предложенные автором методы и инструменты выходят за рамки разработки программного продукта. Книга будет интересна опытным и начинающим руководителям из любой отрасли, линейным менеджерам, студентам, изучающим менеджмент и разработку программного обеспечения.

УДК 658.5:004
ББК 65.291.21с5165.291.21с51

Содержание

Скотт Беркун
Сделано
Проектный менеджмент на практике

Издано с разрешения O'Reilly Media, Inc.
Благодарим за консультации Илону Ноженко

Authorized Russian translation of the English edition of Making Things Happen ISBN 9780596517717

* * *

Предисловие

С первым изданием случилось невероятное. Книга разошлась огромным тиражом; попала в несколько списков бестселлеров, стала номинантом различных премий и привлекла достаточно внимания, побудив автора объехать весь мир и рассказать о своих идеях. А потом произошло еще более неожиданная вещь: у книги поменялось название.

Мы с издательством O'Reilly договорились, что если уж книге предстоит обрести вторую жизнь (изначально книга называлась «Искусство управления IT-проектами»[1]), то нужно ее доработать. Мы обновили и дополнили текст, чтобы чтение доставило вам еще большее удовольствие. Если вам интересно, почему изменили название, позвольте перечислить несколько возможных причин.

1. Министерство национальной безопасности усмотрело в старом названии террористическую угрозу.

2. Тим О'Райли[2] решил, что его медиаимперия станет № 1 в мире, если он пойдет на хитрость со сменой названия и убедит читателей купить книгу во второй раз.

3. А здесь можете записать любое объяснение, какое придет вам в голову.

Какой бы ни была причина, мы снова взялись за старое. Я сделал все, что в моих силах, чтобы улучшить книгу и при этом не обречь ее на фиаско «Звездных войн» Джорджа Лукаса[3]. Перечислим в двух словах, что изменилось.

• Текст вычитан на предмет ясности и краткости изложения. Книга стала более содержательной, без «воды».

• В конце каждой главы добавлено более 120 упражнений, стимулирующих мозговую деятельность.

• Я добавил новый раздел по дискуссионным группам, чтобы помочь вам сформировать такую группу и продолжить обучение.

Если вы никогда не слышали об этой книге, введение расскажет вам все, что нужно знать.

После выхода первого издания два года назад я был очень занят. Я написал другую книгу под названием «Откуда берутся гениальные идеи»[4], выпустил ряд статей, подкастов и видео. Я продолжаю вести популярный блог по креативу и менеджменту. Все это можно найти на www.scottberkun.com.

С наилучшими пожеланиями,
Скотт Беркун
Редмонд, Вашингтон
Март 2008 г.

---

[1] Беркун С. Искусство управления IT-проектами. СПб.: Питер, 2007.

[2] Тим О'Райли (р. 1954) – американский издатель, основатель издательства O'Reilly, активист движения за свободное программное обеспечение и программное обеспечение с открытым исходным кодом; считается одним из главных идеологов Веб 2.0. *Прим. ред.*

[3] Лукас Д., Глут Д., Кан Дж. Звездные Войны. Оригинальная трилогия. М.: Азбука-Аттикус, 2016.

[4] Беркун С. Откуда берутся гениальные идеи? 10 мифов об инновации. СПб.: Питер, 2011.

Введение

* * *

Мой любимый вопрос – «как?». Как это работает? Как это устроено? Как они это сделали? Стоит мне увидеть что-то интересное, меня переполняют вопросы с этим коротким, но сильным словом. И большинство ответов строится на том, как люди применяют свои интеллект и мудрость, а не конкретные технологии и теории.

За годы разработки программных продуктов и сравнения своего опыта с опытом других менеджеров, программистов и проектировщиков я научился эффективно управлять проектами. Эта книга – квинтэссенция всего, что я узнал. Сюда вошли методы управления командами, принципы работы с идеями, организации проектов, планирования графика, разруливания конфликтных ситуаций и достижения результата – даже перед лицом серьезных испытаний и несправедливостей.

Несмотря на несколько обобщенное название книги, мой основной опыт работы связан с технологиями, в частности в Microsoft Corporation. Я проработал там с 1994 по 2003 годы, возглавлял команды по проектам Internet Explorer, Microsoft Windows и MSN. Несколько лет я посвятил группе совершенствования разработок Microsoft. Я обучал и консультировал команды во всей компании, часто выступал на конференциях, читал лекции в других корпорациях и университетах. Большинство советов, уроков и примеров, собранных в книге, опираются именно на этот опыт.

Я всю жизнь занимаюсь разработкой программного обеспечения и приложений, но эту книгу подготовил для широкой аудитории. Я описал методы и инструменты, выходящие за рамки разработки. Думаю, они будут интересны и полезны предпринимателям из любой

отрасли. Создатели тостера, небоскреба, автомобиля, сайта или программных продуктов испытывают схожие проблемы, и эта книга в первую очередь посвящена тому, как их преодолеть.

В отличие от других трудов по управлению проектами она не опирается ни на одну грандиозную теорию или якобы новаторскую философию. Напротив, я сделал акцент на практичность и разнообразие. Успех проекта зависит от правильного распределения сотрудников, сочетания их навыков, отношения к работе и тактики руководства. Послужной список (или его отсутствие) не имеют никакого значения. Как мне кажется, я придумал самую разумную структуру книги: анализ основных типов ситуаций и советы, как выйти из них победителем. Я немало потрудился, чтобы выбрать правильные темы и посоветовать что-то дельное. Надеюсь, это было не зря.

Для кого эта книга

Чтобы узнать, подходит вам эта книга или нет, предлагаю вернуться к оглавлению, выбрать тему, которая вас интересует, и пробежать глазами, что я написал. Я никогда не доверял введениям и вам не советую, но все-таки продолжу.

Книга будет особенно полезна тем, кто может отнести себя к одной или нескольким категориям.

• **Опытные тимлиды и руководители.** Эта книга прекрасно подходит для лидеров проектов любого типа. Несмотря на примеры из области разработки программного обеспечения, идеи легко применимы и к другой работе. Вы можете быть лидером команды по должности или просто одним из более опытных ее членов. И хотя некоторые темы наверняка покажутся вам знакомыми, вы сможете выработать свое мнение, даже если не согласны со мной.

• **Новоиспеченные тимлиды и руководители.** Оглавление представляет собой подробный перечень функционала менеджеров проекта. В каждой главе анализируются самые распространенные ошибки, свойственные даже опытным людям, выясняется, почему они происходят и как их избежать. Вы окинете широким взглядом круг своих обязанностей и эффективные способы их выполнения. Поскольку большинство глав посвящено довольно масштабным темам, они часто содержат ссылки на более подробные источники.

• **Программисты, тестировщики и другие участники проекта.** Эта книга поможет разобраться, какие цели стоят перед вами и какие подходы можно применить для эффективной работы. Если вы когда-либо задумывались, почему проекты так часто меняют направление или создается впечатление, что ими неграмотно управляют, эта книга поможет понять причины и найти решения. Как минимум она увеличит шансы на то, что ваша работа изменит что-то к лучшему (и вы не сойдете с ума в процессе). Если вам хотелось бы возглавить команду, эта книга поможет узнать, как выглядит лидерство и насколько подходит лично вам.

• **Студенты, изучающие менеджмент, проектирование и разработку программного обеспечения.** Я использую слово «студенты» в широком смысле: если вы интересуетесь этими темами или официально учитесь, эта книга должна привлечь ваше внимание. В отличие от того, что можно прочитать в учебниках, мы делаем акцент на конкретику. Опыт и примеры абсолютно реальны, и именно они легли в основу уроков и методов, а не наоборот. Я намеренно избегал разграничения учебных дисциплин: на мой взгляд, разграничения никогда не помогали ни проектам, ни пониманию реального положения дел (мир не делится по тому же принципу, что университетские занятия). Напротив, в этой книге переплетаются вопросы бизнес-теории, психологии, тактики менеджмента, процессов проектирования и разработки ПО таким образом, чтобы из каждой темы читатель смог извлечь для себя пользу.

Что я думаю о вас

• **Вы не глупы.** Надеюсь, я правильно выбрал тематику и хорошо написал, и вы не хотите, чтобы я ходил вокруг да около. Лучше я сразу перейду к делу. Даже если у вас меньше или больше опыта или вы специалист в совершенно другой области, я считаю вас коллегой, обратившимся ко мне за советом.

• **Вы любопытны и прагматичны.** Я опираюсь на примеры из многих дисциплин и предполагаю, что вы найдете для себя ценные уроки, не ограниченные разработкой приложений и программного обеспечения. Для пытливых умов я приготовил ссылки и дополнительные материалы, иногда в сносках, так что они не будут мешать чтению. Предполагаю, вы хотите учиться, открыты новым идеям и понимаете ценность аргументированных мнений, даже если не согласны с ними.

• **Вы не поклонник жаргона и громких теорий.** Не думаю, что жаргон и громкие теории помогают усваивать и применять новую информацию. Я избегаю их, если они не ведут меня к полезным сведениям, которые пригодятся позже.

• **Вы не слишком серьезно относитесь к себе, программному обеспечению и менеджменту.** Разработка программного обеспечения и управление проектами могут наводить тоску. Конечно, такая книга не отличается интригующим сюжетом (другое дело, если бы о разработке программного обеспечения написал Марк Твен или Дэвид Седарис[5]), но я решил не лишать себя удовольствия пошутить над собой (или другими) или разъяснять те или иные моменты на забавных примерах.

[5] Марк Твен (настоящее имя Сэмюэл Лэнгхорн Клеменс, 1835–1910) – всемирно известный американский писатель, журналист и общественный деятель. Его творчество охватывает множество жанров: юмор, сатиру, философскую фантастику, публицистику и другие. Дэвид Рэймонд Седарис (р. 1956) – американский юморист, комик, автор и радиопостановщик. *Прим. ред.*

Как пользоваться этой книгой

Если в какой-то момент вы заскучаете или сочтете примеры неактуальными, пропустите это место. Я написал эту книгу, учитывая потребности людей, которые привыкли «листать», а не читать или же нуждаются в конкретном совете прямо сейчас. С главами можно знакомиться в любом порядке, особенно с теми, что посвящены человеческой природе (главу 8, главу 9, главу 10, главу 11, главу 12, главу 13 и главу 16). Однако есть польза и от чтения по порядку, от начала до конца. Некоторые концепции, представленные позже, опираются на те, что даны раньше, и большинство проектов описываются в хронологическом порядке. Первая глава – самая общая и глубокая. Если вам любопытно, зачем вообще нужен проект-менеджмент или что говорят о нем авторитетные люди, то стоит почитать ее. Если начнете и вам не понравится, настоятельно рекомендую опробовать другую главу, прежде чем покинуть корабль.

Все ссылки и URL, приведенные в книге, а также дополнительные примечания и комментарии, доступны онлайн на www.makingthingshappen.org. Если вас интересуют дискуссионные группы по этой книге, загляните в приложение. Там вы узнаете, какие группы существуют и как организовать свою собственную группу.

Раз вам хватило терпения прочитать введение целиком, предполагаю, вы прекрасно разбираетесь и в других аспектах оформления книги (нумерация страниц, ссылки и так далее), и мне пора наконец оставить вас в покое.

Глава первая. Краткая история управления проектами (и зачем это вам нужно)

* * *

Во многих компаниях руководитель проектов не всегда занимает одноименную должность. И в этом нет ничего страшного. Все так или иначе управляют проектами, работая самостоятельно или руководя командой. Сейчас эти различия не существенны. Моя задача – выяснить, что делает проект успешным и как люди, которые управляют успешными проектами, добиваются цели. Эти стратегии не требуют особой иерархии, должностей и методов. Итак, если вы работаете над проектом и хотя бы в некоторой степени ответственны за его результат, все дальнейшее касается именно вас. И если на вашей визитке указано *менеджер проекта*, тем лучше.

Эту книгу можно рассматривать с трех позиций: как сборник отдельных статей, как цельное повествование и как справочник по наиболее распространенным ситуациям. Каждая глава посвящена отдельной высокоуровневой задаче, намечает общий план работы и предлагает методы ее успешного выполнения. Однако в первой главе я придерживаюсь иного подхода: есть три более общих момента, которые облегчат дальнейшее чтение, и мне бы хотелось обсудить их прямо сейчас.

Первый – краткая история проектов и почему нужно учиться на том, что делали другие. Второй – история разных типов управления проектами и примеры из моего опыта работы в Microsoft. И третий – обзор основных трудностей, и как их преодолеть. Несмотря на свою

пользу, эта информация не потребуется для понимания последующих глав. Так что, если первая глава кажется вам слишком общей, смело переходите ко второй.

История нам в помощь

Проект-менеджмент как идея зародился очень давно. Если вспомнить все, что создала наша цивилизация, то у нас тысячелетний опыт проектов, на котором можно учиться. От современных разработчиков ПО ниточка тянется к строителям египетских пирамид или римских акведуков. В любую эпоху менеджеры проекта играли одну и ту же роль: применяли технологии к решению актуальных для своей эпохи задач. Однако сегодня, пытаясь улучшить управление проектами и разработку ПО, многие пренебрегают уроками прошлого. Как правило, в поисках полезных знаний мы выбираем из всей линии времени отрезок, приближенный к сегодняшнему дню.

История инженерных проектов показывает, что большинство из них имеет ярко выраженные общие черты. У них есть требования, план и ограничения. Они зависят от умения общаться, принимать решения, сочетать креатив и логику. Как правило, проекты предполагают определенный график работы, бюджет и клиентов. А главное, основная задача проекта – объединить работу разных людей в единое целое и создать нечто полезное для других (клиентов). Мы можем использовать HTML, C++ или бетон и сталь, но для всех проектов существуют общие неизменные принципы.

Я пришел к ним, интересуясь эффективными методами разработки приложений и программного обеспечения. Обратил внимание на другие области, чтобы узнать, как там решаются важнейшие задачи и проблемы проектов. Стал разбираться в организации таких проектов, как космический телескоп «Хаббл» и «Боинг 777». Можно ли заимствовать что-то из их спецификаций и планирования? Есть ли сходство в управлении проектами между моими программистами и строителями афинского Парфенона или небоскреба «Крайслер-билдинг» в Нью-Йорке? А в чем отличия и чему учит их анализ?

Возьмем, к примеру, редакторов газет, которые ежедневно «производят» информацию. Ведь они занимались мультимедиа (иллюстрациями и словами) задолго до того, как появилась идея публикаций. А съемки художественного фильма? Запуск «Аполлона-13»? Изучая эти вопросы, я смог найти новые способы управления проектными командами.

Однако ответы не всегда очевидны. Не могу обещать, что вы быстрее выпустите продукт или научитесь лучше планировать благодаря этим источникам информации. Однако я точно знаю, что, вернувшись в мир программного обеспечения после путешествия в далекие области знаний, я по-новому взглянул на собственные процессы и инструменты. Многие полезные подходы и сравнения, которые мне удалось отыскать, никогда не упоминались на лекциях по информатике в колледже. Их не обсуждали на конференциях технологического сектора и о них никогда не писали в отраслевых журналах.

Перечислю ключевые выводы своих исторических исследований.

1. Управление проектами и разработка ПО – не сакральные знания. Любая современная инженерно-техническая деятельность – лишь очередной вклад в многовековую историю созидания и творения. Технологии и навыки меняются в отличие от ключевых проблем инженеров. Все задачи, будь то языки программирования или методы разработки, уникальны в том или ином смысле, однако являются производной других задач. И если мы хотим позаимствовать как можно больше знаний из прошлого, нужно быть готовыми исследовать оба аспекта – уникальность и преемственность – и сравнивать свои задачи с тем, что было раньше.

2. Чем проще вы воспринимаете свои задачи, тем эффективнее и сосредоточеннее выполняете работу. Простой взгляд на свою деятельность поможет найти больше примеров и уроков из истории и современных отраслей, позаимствовать их, найти сходства или

различия. Это похоже на японскую концепцию *шошин*, что означает «сознание новичка»[6], или открытый разум, – важную часть многих боевых искусств. Пытливый и открытый ум делает возможным рост и требует немало практики. Познание нового требует выйти за узкие рамки «безопасного» восприятия своей работы.

3. Простой не значит легкий. Ведущие спортсмены, писатели, программисты и менеджеры воспринимают свои занятия как нечто простое и при этом сложное. Помните, что простой и легкий – разные вещи. К примеру, пробежать марафон – простая задача. Вы начинаете бежать и не останавливаетесь, пока не преодолеете 42 километра. Куда уж проще? То, что это нелегко, не отменяет простоты задачи. Быть лидером и менеджером тоже сложно, но по своей природе делать работу определенным образом ради достижения определенной цели – это просто.

Я буду возвращаться к этим концепциям во многих главах. Так что, если я ссылаюсь на примеры, которые выходят за рамки стереотипов разработки программного обеспечения, надеюсь, вы поймете почему. И когда я пишу, что принимать решение или планировать график – простые функции управления, я ни в коем случае не имею в виду, что их легко осуществить.

УЧИМСЯ НА ОШИБКАХ

> Человеческие существа отличаются практически уникальной [среди животных] способностью учиться на чужом опыте, но при этом удивляют очевидным нежеланием делать это.
> *Дуглас Адамс*[7]

Мои исследования поставили меня перед вопросом: зачем добровольно страдать от ошибок и разочарований, когда можно избежать их? Если история древнего и современного управления проектами известна и нам платят за умные решения, независимо от источника вдохновения, почему так мало компаний вознаграждает людей за то, что они обращаются к урокам прошлого? Проекты выполняются или закрываются (а именно такая судьба ждет многие проекты[8]), но причины этого анализируются редко. Создается впечатление, что менеджеры большинства организаций не вознаграждают людей за подобную информацию. Возможно, они боятся того, что можно обнаружить (и ответственности). Или никому не интересно анализировать неприятный, плачевный опыт, когда можно потратить время на новую задачу.

В своей книге «Инжиниринг и человеческий фактор: роль ошибки в успешном дизайне» (To Engineer Is Human: The Role of Failure in Successful Design, Vintage Books, 1992) Генри Петроски[9] объясняет, что ошибки нередко приводили к прорывам в разработках. Отчасти это происходит потому, что ошибки вынуждают нас быть внимательнее. Они требуют, чтобы мы пересмотрели предположения и принципы, о которых забыли (сложно притворяться, что все хорошо, когда прототип горит синим пламенем). Как говорил Карл Поппер[10], есть

---

[6] Сознание новичка – одна из основных концепций дзен-буддизма. Традиционный пример – стакан: если цепляться за его содержимое, в нем никогда не будет достаточно места для новых знаний. См. Shunryu Suzuki Zen Mind, Beginner's Mind (Weatherhill, 1972) (Судзуки С. Сознание дзен, сознание начинающего. М.: Альпина Паблишер, 2014.).

[7] Дуглас Ноэль Адамс (1952–2001) – английский писатель, драматург и сценарист, автор юмористических фантастических произведений. Известен как создатель знаменитой серии книг «Автостопом по галактике». *Прим. ред.*

[8] The CHAOS Report (The Standish Group) – документ о бюджете, графике работы и общих ошибках и неудачах проектов по программному обеспечению. См. http://standishgroup.com/sample_research/.

[9] Генри Петроски (р. 1942) – американский инженер, кандидат теоретической и прикладной механики. Профессор гражданского строительства и истории в Университете Дьюка. Автор книг, в которых подробно описывается история промышленного дизайна обычных повседневных предметов, таких как карандаши, скрепки и столовое серебро. *Прим. ред.*

[10] Карл Поппер – известный философ и социолог XX века. См. http://en.wikipedia.org/wiki/Karl_Popper.

только два типа теорий: ошибочные и недоработанные. Без неудач мы самонадеянно забываем, что наше понимание несовершенно.

Смысл в том, чтобы учиться на чужих ошибках. Хотя детали провалов могут быть разными в зависимости от проекта, основные причины или ошибочные действия команды вполне можно применить к вашему случаю (и избежать их). Пора прекратить прятаться от поражений. Напротив, нужно рассматривать их как возможность чему-то научиться. Какие факторы стали причиной неудачи? Какие из них можно свести к минимуму или полностью исключить? Согласно Петроски, знания, которые мы черпаем из неудач, – самый богатый источник прогресса, если нам хватит смелости проанализировать, что произошло.

Возможно, именно поэтому в Boeing Company, одном из крупнейших в мире производителей авиационной техники, конструкторские просчеты становятся уроками[11]. Boeing ведет свою черную книгу с момента основания компании и помогает инженерам извлекать опыт из ошибок прошлого. Поступая так, любая компания не только повышает шансы на успех проектов, но и создает условия для открытого обсуждения неудач, отрицает их или прячется. Мне кажется, разработчикам ПО тоже стоит вести черные книги.

[11] James R. Chiles, Inviting Disaster: Lessons from the Edge of Technology (HarperBusiness, 2002).

Разработка, кухня и скорая помощь

Есть одна проблема: иногда невозможно применить уроки двадцатилетней давности и интересоваться тем, что так сильно отличается от нынешних реалий. Единственный выход – провести аналогии с современными проектами. Такому подходу будет недоставать предыстории разработок, а потому веса и авторитета, но все же он позволяет познакомиться с личным опытом и наблюдениями людей. Зачастую увидеть все своими глазами – единственный способ собрать достаточно информации, чтобы провести аналогии между различными идеями.

Один мой знакомый разработчик уверен, что коль скоро принимает сложные решения – по проектированию, координации процесса, тестированию изменений за несколько часов или даже минут, затем выпуску продукта, – его проектам и обязанностям нет аналога во всей истории Вселенной. Он с гордостью перечислит все, чем овладел: CSS, XHTML, Flash, Ajax и другое – и станет убеждать, что пятьдесят лет назад эти технологии потрясли бы величайшие умы. Уверен, вам тоже доводилось встречать таких людей. Или, возможно, самим казалось, что до вас никто не решал такие сложные задачи.

Я предложил этому разработчику заглянуть «за кулисы» его любимого ресторана в разгар рабочего дня. Всем будет весьма полезно хоть раз получить представление о том, что творится на кухне (рекомендую блестящую книгу Энтони Бурдена «О еде: строго конфиденциально»[12]), однако сейчас я подразумеваю продуктивность. Когда люди впервые видят молниеносное управление и координацию работы на профессиональной кухне в разгар рабочего дня, им уже не кажется, что их работа самая сложная на свете. Повара жонглируют горячими сковородками с разными блюдами на разном этапе готовности, носятся между конфорками из одного угла кухни в другой, а официанты при этом вбегают и выбегают, сообщают о новых заказах, изменениях и проблемах.

И все это происходит в небольшом тесном помещении, где температура намного выше 30 градусов, под ярким флуоресцентным светом. И сколько бы заказов они ни готовили каждые несколько секунд, новые поступают примерно с той же скоростью. Иногда заказы возвращаются обратно, или, как в программных проектах, в последнюю минуту клиент вносит поправки (столик 1 не переносит лактозу, на столик 2 нужен соус и т. д.). Наблюдать за большими шумными кухнями – одно удовольствие. Хотя на первый взгляд кажется, что там царит полнейший хаос, большинство команд разработчиков только позавидует интенсивности и точности действий лучших кухонь.

Шеф-повара и линейные повара – менеджеры кулинарных проектов или, как называет их Бурден, регулировщики движения (кстати, еще одна профессия для вдумчивого изучения). Хотя персонал кухни работает в гораздо меньших масштабах и пользуется гораздо меньшим почетом, чем менеджер команды разработчиков, по ежедневной интенсивности их даже сравнивать нельзя. Если не верите, когда в следующий раз окажетесь в ресторане, спросите официанта, можно ли одним глазком посмотреть на работу кухни. Скорее всего, он откажет, но если все-таки согласится, вы не будете разочарованы. (В некоторых модных ресторанах и барах есть открытая кухня. Если найдете такую, сядьте поближе и понаблюдайте за одним из поваров хотя бы несколько минут. Смотрите, как размещаются заказы, как они отслеживаются, собираются и доставляются. Если на кухне аврал, то вы совершенно по-другому посмотрите на поиск и исправление ошибок в программах.)

Еще одна любопытная область – скорая помощь. Я смотрел по каналам Discovery и PBS, как небольшие команды врачей и медсестер оказывают пациенту помощь и находят выход из совершенно непостижимых ситуаций. Не удивительно, что именно врачи неотложной помощи

[12] Бурден Э. О еде. Строго конфиденциально. Записки из кулинарного подполья. М.: Мидгард Эксмо, 2011.

изобрели процесс триажа[13], который применяется в разработке ПО для сортировки задач или ошибок в соответствии с приоритетами (подробнее – в главе 15).

В медицинской среде, особенно травматологии, находятся удивительные параллели для командной работы, принятия решений в условиях высокого стресса и результатов проекта, которые влияют на многих людей каждый день (на рисунке 1.1 вы видите краткое сравнение с медициной и другими сферами работы). Как писал Атул Гаванде[14] в своей блестящей книге «Сложности: заметки хирурга о несовершенной науке» (Complications: A Surgeon's Notes on an Imperfect Science (Picador USA, 2003):

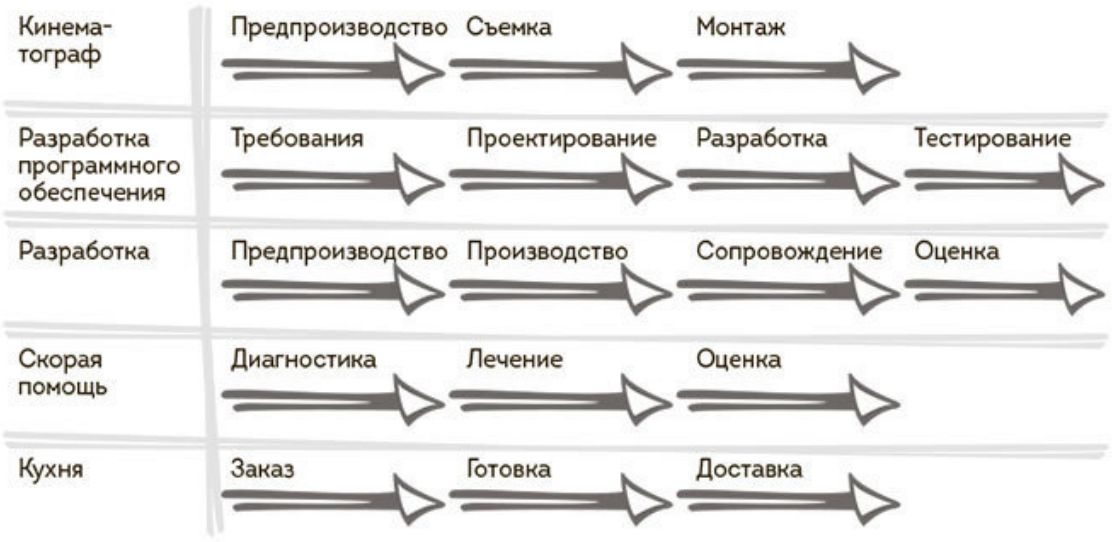

Рис. 1.1. В теории многие дисциплины опираются на схожие процессы. Все они уделяют время планированию, исполнению и совершенствованию. (Однако не стоит обращаться на кухню за медицинской помощью или обедать в отделении реанимации.)

> Мы воспринимаем медицину как упорядоченную область знаний и четкую процедуру. Однако это не так. Медицина – наука несовершенная, собрание постоянно меняющихся сведений, непроверенной информации, небезупречных людей – и жизней, которые нужно спасти. Да, наша работа опирается на науку, но, помимо нее, – на привычки, интуицию, а иногда и просто догадки. Между нашими знаниями и стремлениями остается чудовищная пропасть. И эта пропасть усложняет все, что мы делаем.

Этот принцип, как и многие другие идеи из блестящей книги Гаванде, применим к разработке программного обеспечения. Фред Брукс[15] в своей классической работе по софтверному инжинирингу «Мифический человеко-месяц»[16] проводит схожие сравнения между командами хирургов и программистов. Хотя жизнь редко стоит на кону, когда вы работаете над сайтом или базой данных, можно отметить немало обоснованных параллелей с трудностями, которые подстерегают эти команды абсолютно разных профессионалов.

---

[13] Триаж (фр. Triage – сортировка) – распределение пострадавших и больных на группы, исходя из нуждаемости в первоочередных и однородных мероприятиях (лечебных, профилактических, эвакуационных) в конкретной обстановке. *Прим. ред.*

[14] Гаванде А. (р. 1965) – американский хирург, журналист, писатель. Широко известен как эксперт в области оптимизации современного здравоохранения. *Прим. ред.*

[15] Фредерик Филлипс Брукс – младший (р. 1931) – американский ученый в области теории вычислительных систем. Управлял разработкой OS/360 в IBM. Награжден премией Тьюринга в 1999 году. *Прим. ред.*

[16] Брукс Ф., Чапел Х. Мифический человеко-месяц, или Как создаются программные системы. СПб.: Символ-Плюс, 2010.

Задача проектного менеджмента

Управление проектами может быть профессией, должностью, ролью или функцией. В некоторых компаниях есть руководители, которые отслеживают все проекты двухсот сотрудников. В других этой должностью «награждают» линейных младших менеджеров, каждый из которых реализует часть масштабного проекта. В зависимости от структуры организации, ее культуры и целей проекта управлять проектами может любой сотрудник одновременно с выполнением других обязанностей или сотрудник, для которого управление будет основной задачей (Винсент, Клод и Рафаэль – наши постоянные менеджеры проекта).

В этой книге я использую термин *менеджер проекта* (project manager) применительно ко всем, кто *управляет проектом*. Под таковым я подразумеваю руководство командой на этапе планирования, выстраивания графика и формулировки требований, а также на этапах проектирования, разработки (коммуникация, принятие решений и стратегия «середины игры») и достижения итогового результата (лидерство, кризис-менеджмент и конечная стратегия).

Если в вашей сфере эти функции не формализованы, под менеджером проекта можно подразумевать «человека, который управляет проектом / руководит общим ходом проекта, даже если это не входит в его основные обязанности». Я встречал разные формы этих функций, распределенных среди всех членов команды, и скажу, что это несущественно. Эта книга не о должностях и формальностях, а о том, как реализовать свои цели и добиться результата. Однако чтобы максимально упростить вам чтение, я остановлюсь на термине *менеджер проекта*, или *МП*.

Иногда отсутствие менеджера проекта не бросается в глаза. Программисты и их начальство составляют графики и планы работы (если они есть), а бизнес-аналитик или маркетолог занимается планированием и требованиями. Все остальные функции, которые можно отнести к проект-менеджменту, просто распределяются по членам команды. Возможно, этих людей наняли как раз благодаря их интересам, помимо написания кода. Возможно, они вовсе не против заниматься первичным планированием, UI-дизайном или бизнес-стратегией. В такой рабочей структуре заложена возможность значительной оптимизации. Раз все готовы нести ответственность за результат и равномерно распределять ношу, которую отдельный МП целиком и полностью взвалил бы на свои плечи, то в команде будет на одного сотрудника меньше. Эффективность и простота – всегда хорошо.

Но отсутствие МП порой чревато серьезными проблемами. Личные интересы одного члена команды могут легко сбить с толку остальных, если не появится кто-то, способный взять под контроль всю работу команды. Среди проектировщиков и бизнес-аналитиков формируются враждебные фракции, замедляя прогресс и препятствуя работе других членов команды.

В больнице один из врачей несет ответственность за лечение пациента. Это ускоряет процесс принятия решений и дает четкое представление об обязанностях каждого сотрудника отделения. Команда разработчиков без руководства может попасть в довольно затруднительное положение в случае проблем. Если никто не проводит триаж ошибок или не следит за графиком работы и не отмечает проблемные моменты, это серьезно повлияет на результат.

Многие программисты достаточно хорошо разбираются в проект-менеджменте, чтобы самим заниматься этим, и признают уникальную ценность профессионального, увлеченного человека, который готов взять на себя роль лидера.

Программный и проектный менеджмент в Microsoft

В конце 1980-х годов Microsoft столкнулась с проблемой координации технической работы с задачами маркетинга и бизнеса (некоторые скажут, что это до сих пор остается проблемой для Microsoft и многих других компаний). Мудрый человек по имени Джейб Блюменталь придумал должность, на которой человек одновременно будет выполнять лидерские и организационные функции. Ему предстояло работать над проектом с первого дня планирования и до последнего дня тестирования. Он должен был разбираться в технических вопросах настолько, чтобы завоевать уважение программистов, уметь широко смотреть на продукт, а кроме того с радостью заниматься написанием спецификаций, анализом маркетинговых планов, составлением графиков, управлением командами, стратегическим планированием, триажем ошибок, мотивированием команды и любой другой необходимой работой, которую не выполняет никто иной (по крайней мере, достаточно хорошо). Новая должность в Microsoft получила название *программный менеджер*. Он обладал значительными полномочиями, хотя не все члены команды находились в его прямом подчинении. (В теории менеджмента это пример матричной организационной структуры[17], которая предполагает двойное подчинение: в зависимости от должности сотрудника и в зависимости от проекта. Так, программист или тестировщик подчиняются двум менеджерам.)

Джейб стал программным менеджером в работе над продуктом под названием Multiplan (позже его переименовали в Microsoft Excel), и успех не заставил себя ждать. Улучшились процессы проектирования и разработки и координация работы команды, и все были счастливы. После многочисленных совещаний и рабочих встреч команды адаптировались к новой роли. Поручив конкретные обязанности линейному эксперту широкого профиля, Microsoft навсегда изменила динамику работы команд разработчиков. Практически все время работы в Microsoft я как раз выступал в роли программного менеджера. Я работал с продуктовыми командами Internet Explorer, MSN и Windows и даже руководил командами программных менеджеров.

И по сей день я не смогу назвать много компаний, которые бы решили переосмыслить и формализовать отдельную форму проектного менеджмента. В своих многочисленных взаимодействиях с другими фирмами, которые занимались разработками и софтом, я редко встречал человека, который исполнял бы схожие обязанности (либо это были разработчики, либо специалисты по маркетингу, либо, в редких случаях, проектировщики). Многие компании используют командную структуру для организации работы, однако редко кто выделяет роли, которые совмещают функции проектирования и бизнеса. Сегодня в Microsoft трудятся более 5000 программных менеджеров (из 80 000 сотрудников), и хотя изначальный смысл этой концепции уже давно исказили, многие команды еще придерживаются основных принципов.

Однако независимо от того, что было написано на моей визитке или каким легендам Microsoft вы верите, мои ежедневные обязанности в качестве программного менеджера совпадали с функциями менеджера проекта. В двух словах, это означало, что я ответственен за успех проекта – и всех, кто в нем участвует. Все главы этой книги отражают основные задачи, связанные с этой функцией, от предварительного планирования (глава 3 и глава 4) до спецификаций (глава 7), проектных решений (глава 8), разработки и выпуска (глава 14 и глава 15).

Помимо перечисленных навыков, пригодятся определенные личные характеристики и отношение к работе – иначе каждый, кто руководит или управляет проектом, окажется в невыгодном положении.

---

[17] Резюме матричной организационной структуры и других типов см. Steven A. Silbiger The Ten-Day MBA (William Morrowand Company, 1993) (Силбигер С. MBA за 10 дней. Самое важное из программ ведущих бизнес-школ мира. М.: Альпина Паблишер, 2018), pp. 139–145. Хотя информацию можно найти почти в каждой книге по теории менеджмента.

Сбалансированность проектного менеджмента

Сложно найти хороших менеджеров проекта, потому что им нужно гармонично сочетать в себе разные мировоззрения. В своем эссе Pursuing the Perfect Project Manager[18] Том Питерс называет их парадоксами или дилеммами. Это подходящее название, потому что разные ситуации требуют разного поведения. Менеджер проекта должен не только осознавать все необходимые черты характера, но и выработать в себе чутье, чтобы чувствовать, какие из них приемлемы в тех или иных случаях. Такой взгляд наталкивает на мысль о том, что проект-менеджмент – все-таки искусство: оно требует интуиции, суждения и опыта, чтобы эффективно применять все эти умения. Вкратце перечислим характеристики из эссе Питерса.

• **Эгоизм / альтруизм.** Из-за колоссального объема ответственности менеджеры проекта зачастую черпают огромное личное удовлетворение в работе. Как вы понимаете, они вкладывают немало эмоций в свое дело, и зачастую именно это позволяет им выдержать его интенсивность. Однако в то же время менеджеры проекта не должны ставить личные интересы превыше проекта. Они должны быть готовы делегировать важные или интересные задачи и делить славу со всей командой. Хотя эго может быть полезным топливом, грамотный менеджер проекта должен чувствовать, когда он перегибает палку.

• **Авторитарность / делегирование полномочий.** В некоторых ситуациях самое важное – четкое руководство и быстрый отклик. Менеджер проекта должен быть достаточно уверенным и жестким, чтобы контролировать работу и навязать определенные решения команде. Однако общая цель – избегать подобных крайностей. Проект должен создавать условия, где полномочия можно делегировать и эффективно сотрудничать.

• **Терпимость к неопределенности / стремление к завершенности.** Ранние этапы любого проекта – абсолютно открытый и изменчивый опыт, где неизвестное перевешивает известное. Как мы обсудим в главе 5 и главе 6, контролируемая неопределенность необходима для поиска хороших идей, и менеджер проекта должен чтить эту особенность, даже если не удается управлять ею. Однако есть моменты, особенно на более поздних этапах проекта, когда дисциплина и точность имеют первостепенное значение. Нужна мудрость, чтобы понять, когда стремление к завершенности оправдано, а когда вполне достаточно решения на скорую руку (раздел «Поиск и оценка вариантов» в главе 8).

• **Устное / письменное общение.** Хотя большинство софтверных организаций делает основной акцент на электронную почту, устные навыки общения критически важны. Всегда будут собрания, переговоры, обсуждения в кулуарах и мозговой штурм. МП должен уметь воспринимать и выдавать идеи в личном общении. Чем масштабнее организация или проект, тем важнее становятся навыки письменного общения. Несмотря на личные предпочтения, менеджер должен понимать, в каких случаях эффективнее письменная форма общения, а в каких случаях – устная.

• **Признавать сложность / отстаивать простоту.** Многие становятся жертвой сложности. Столкнувшись с организационными или инженерными трудностями, они тонут в частностях и забывают об общей картине. Некоторые предпочитают отрицать сложность и принимать неудачные решения, потому что плохо воспринимают все тонкости ситуации. Баланс состоит в том, чтобы понимать, какой подход к проекту принесет больше пользы для текущей проблемы, и умело переключаться между разными подходами или держать их в голове одновременно (только чтобы голова не взорвалась при этом). Менеджеры проекта должны убедить команду стремиться к простоте в работе, не приуменьшая при этом всю сложность, связанную с созданием правильного, надежного кода.

---

[18] http://www.tompeters.com/col_entries.php?note=005297&year=1991.

• **Нетерпение / терпение.** Б*о*льшую часть времени менеджер проекта подталкивает команду к действиям, следит за их рациональностью и целенаправленностью. Однако в некоторых ситуациях нетерпение вредит проекту. Периодически приходится тратить время на политические, межорганизационные или бюрократические действия – кто-то должен присутствовать на встречах или в конференц-зале и проявлять терпение. Знать, когда поднажать, а когда отступить, чтобы все шло своим чередом, – способность, которую нужно развивать менеджеру проекта.

• **Отвага / страх.** Одно из величайших заблуждений американской культуры в том, что смелые люди не знают страха. Смелые люди – те, кто испытывает страх, но при этом не опускает руки. Менеджер проекта должен мыслить здраво и понимать, что многое может пойти совсем не так, как планировалось. Однако он должен сочетать это понимание с отвагой, которая необходима, чтобы брать на себя трудные масштабные задачи и решать их.

• **Вера / скептицизм.** Нет ничего лучше для морального духа команды, чем авторитетный лидер, который верит в то, что она делает. Для менеджера проекта важно чувствовать уверенность в работе и видеть истинную ценность в задачах, которые нужно выполнить. В то же время необходима здоровая доля скептицизма (не цинизма) относительно того, как продвигается работа и как решаются эти задачи. Кто-то должен зондировать почву, задавать вопросы, обличать существующие предположения и проливать свет на сложные моменты. Баланс заключается в том, чтобы активно задавать вопросы и бросать вызов убеждениям команды, но при этом не поколебать ее веру в то, что она делает.

Как подчеркивает Питерс, найти людей, обладающих всеми перечисленными качествами и к тому же способных грамотно сочетать их, совсем нелегко. Многие ошибки, неизбежные для любого МП, связаны с неверным сочетанием одной или нескольких конфликтующих сил. Однако любой может совершенствовать их баланс. Я не буду больше останавливаться на этом списке парадоксов (хотя несколько раз мы его еще упомянем), однако ему стоит уделить внимание. Этот список конфликтующих, но при этом необходимых качеств поможет вам пересмотреть, что вы делаете и почему, и принимать более грамотные решения.

Стресс и отвлекающие факторы

Один из страхов новоиспеченных менеджеров проекта заключается в том, что успех требует изменений. Новые проекты создаются с намерением изменить мир, модифицируя, созидая или разрушая что-то. Сохранение статус-кво – если, конечно, это не основная цель по какой-либо непостижимой причине – не самый успешный результат. Мир меняется постоянно. Если сегодня проект не настолько успешен, как в прошлом году, значит, либо задачи были сформулированы некорректно, либо реализация оказалась не на высоте, и он больше неактуален.

Именно поэтому МП постоянно испытывают стресс – это неотъемлемая часть работы. Не сидите без дела, улучшайте мир! Всегда есть новые методы мышления, темы для исследования и применения или процесс, который делает работу более приятной и эффективной. Возможно, за новые внедрения ответственны лидеры, а не менеджеры, однако разница между ними несущественна. Сколько бы мы ни пытались разделить эти две функции, менеджмент, безусловно, требует лидерских качеств, а лидерство – менеджерских. Любой, кто управляет проектами, обязан исполнять обе роли независимо от того, что сказано в должностных обязанностях.

Однако, возвращаясь к стрессу, должен сказать, что многие менеджеры уклоняются от лидерских функций (например, когда команде или проекту нужен человек, который предпримет решительное действие) и предпочитают отслеживать работу команды, вместо того чтобы мотивировать ее или даже самому участвовать в работе. Если человек только контролирует и наблюдает со стороны, возможно, ему больше подходит работа в бухгалтерии. Если лидер в ответ на давление и стресс устраняется, он не лидер – он прячется. Неэффективные менеджеры имеют тенденцию растворяться на периферии проекта, где не приносят практически никакой ценности.

НЕ ПУТАЙТЕ ПРОЦЕСС И ЦЕЛИ

В описанной выше ситуации некоторые менеджеры проекта начинают подсчитывать то, что совершенно не нужно подсчитывать. Не представляя, чем еще заняться, или опасаясь делать необходимое, они тратят время на второстепенные задачи. И по мере того как растет пропасть между МП и проектом, объем ненужного внимания, уделяемого графикам, таблицам, чек-листам и отчетам, тоже увеличивается. Вероятно, в какой-то момент менеджер начинает думать, что данные и процесс – это и есть проект. Он сосредоточивается на менее важных и более легких задачах (таблицы и отчеты) вместо тех, с которыми тяжело работать (программирование или график). Такой человек убежден, что достаточно следовать определенной процедуре, довести ее до совершенства, поставить галочки по всем пунктам чек-листа – и проекту гарантирован успех. Или избирается еще более циничный подход: любая неудача, которая может произойти, с технической точки зрения перекладывается на чужие плечи.

Чтобы свести к минимуму возможную путаницу, грамотные менеджеры не делят работу на приятную и неприятную. Они не проводят границ между задачами проект-менеджмента и самим проектом. Жесткая привязанность к чек-листу предполагает наличие определенного процесса, который гарантирует определенный результат, а так никогда не бывает. В реальности есть три вещи: задача, масса работы и группа людей. Четко определенные роли (глава 9) помогают организовать работу, однако формирование ролей не самоцель. Чек-лист помогает выполнить работу и достичь цели, однако он тоже не цель. Путать процессы с задачами – одна из самых серьезных ошибок. Знаю, потому что сам грешен.

Много лет назад я руководил разработкой нескольких сложных компонентов пользовательского интерфейса в проекте Internet Explorer 4.0. Я чувствовал себя под прессом: такой важной задачи мне еще никогда не поручали. Я решил, что если все записать в виде чек-листов,

успех гарантирован. Конечно, работу по проекту нужно отслеживать, но я дошел до крайности. Данные отображались в сложной электронной таблице, а в офисе висела огромная доска со списками и таблицами (при этом я собирался повесить еще несколько).

Мой босс не возражал (потому что работа шла вполне успешно), пока не заметил, что я больше времени уделяю чек-листам и процессам, чем команде – это явный сигнал тревоги. Однажды босс зашел в мой офис и, взглянув на необъятные матрицы чек-листов и таблиц, которые красовались на каждой плоской поверхности, закрыл дверь, усадил меня и сказал: «Скотт, все это очень мило, но проект – это в первую очередь команда. Займись ею, а не чек-листами. Если они помогают тебе руководить, это зд*о*рово. Однако учитывая твой настрой, тебе скоро понадобится помощь команды, чтобы управиться с чек-листами».

Итак, вместо акцента на процессах и методах менеджер проекта должен сосредоточиться на команде. Простой план и система мониторинга, конечно, необходимы, но они должны соответствовать сложности проекта и культуре команды. Точнее, планирование и мониторинг должны поддерживать команду на пути к достижению целей проекта, а не мешать ей. Я уверен, что пока МП внимателен и пользуется уважением сотрудников, любые недостающие задача, процесс, отчет, чек-лист или другие необходимые инструменты станут очевидны до того, как возникнут серьезные трудности.

Как мы обсудим в главе 10, если книга или руководитель велят что-то делать или если тот же метод использовался в прошлом месяце либо в прошлом году, это еще не значит, что его следует применять сегодня. Каждая команда и проект разнятся. Зачастую можно найти немало веских причин, чтобы подвергнуть сомнению старые принципы и убеждения. Методы и процессы нужно выбирать крайне осторожно, имея в виду, что они разрастаются и затягивают команду в «смоляную яму сложных проектов», как говорится в книге Фреда Брукса «Мифический человеко-месяц». Когда одни процессы нужны для управления другими процессами, сложно понять, где выполняется реальная работа. Зачастую именно МП обладает возможностью избавить команду от бюрократии или же, напротив, толкнуть ее на бесконечный круг процедур и «комитетного» мышления.

Грамотная вовлеченность

Все менеджеры – от управляющих Fortune 500 до тренеров спортивных команд – подвержены чрезмерной вовлеченности. Они понимают, что чаще всего зря сотрясают воздух, и маниакальное участие – один из наиболее удобных (хотя и негативных) способов компенсировать свое бессилие. Это отчасти объясняет бесконечный поток микроменеджеров; самый простой ход для слабого менеджера – злоупотреблять своей властью над подчиненными (и в крайних случаях обвинять их в некомпетентности, из-за которой приходится уделять им столько внимания). Неуверенность, которую испытывают менеджеры, вызвана тем фактом, что, выражаясь терминами индустриальной революции, они не участвуют в производственной цепочке. Они ничего не делают своими руками и по своей ценности явно отличаются от тех, кто делает.

Лидеров и менеджеров нанимают не для линейной работы, как программистов. Напротив, они нужны для того, чтобы увеличить ценность каждого члена команды. Методы увеличения ценности отличаются от производственной работы. Однако поскольку многие менеджеры – бывшие программисты и вышли как раз из сферы производства, велика вероятность того, что у них больше уверенности и навыков для написания кода, чем для управления теми, кто пишет коды.

Как тренер бейсбольной команды, менеджер должен привнести в команду нечто совершенно иное, чем каждый отдельный сотрудник. Например, решить спорные моменты или защитить команду в конфликтных ситуациях; предоставить грамотный, высокоуровневый план или найти неожиданный выход. Поскольку этот вклад сложнее измерить, многие МП мучаются от двойственности своей роли. Как менеджеры, они – удобная мишень для обвинений и прятаться им некуда. Нужны убежденность, уверенность и понимание ситуации, чтобы быть эффективным и счастливым тимлидом.

ВАШЕ ВЫГОДНОЕ ПОЛОЖЕНИЕ

Лучший способ твердо стоять на ногах – воспользоваться психологическим преимуществом некоей отстраненности. В силу своих обязанностей МП общается с разными членами команды чаще, чем кто-либо другой, и в силу этого у него больше источников информации и более точное понимание проекта. МП имеет равное представление о бизнес-ценности и технической стороне проекта и при необходимости может разъяснить все это команде. Общий взгляд на проект позволяет обеспечивать нужных сотрудников критически важными данными в нужное время. Хотя эти полномочия могут иметь довольно широкие последствия, в качестве наглядной иллюстрации приведу простой пример.

У меня была привычка ходить по коридорам и заглядывать к программистам. Обычно я обменивался с ними парой слов, рассказывал анекдот и просил показать, над чем они работают. Иногда смотрел демо-версию. Я делал это раз в два-три дня, тратил всего несколько минут и прекрасно представлял себе реальный статус проекта (в главе 9 мы обсудим этот принцип управления – метод хождения).

Однажды утром, во время работы над проектом IE 5.0, я заглянул в офис Фреда. Обнаружив проблемы совместимости, он спорил со Стивом, другим программистом, о том, как исправить элемент управления для прокручиваемого списка. Никто из них не хотел заниматься этой работой. Не вмешайся я, они бы впустую потратили полдня или даже больше. Я уточнил, о чем спор. Оба замотали головами, словно хотели сказать: «А тебе-то какая разница?» Я предложил им поговорить с Биллом из соседнего отдела. Они снова спросили: «Зачем?», будучи уверены, что мне недоступны тонкости архитектуры проекта. Я улыбнулся и сказал: «Посмотрите на

новый элемент управления, который уже прекрасно работает. Билл столкнулся с проблемой вчера вечером и решил ее в рамках другой задачи».

Конечно, я не спас мир и не предотвратил катастрофу. Если бы я не направил их к нужному человеку, они потратили бы всего несколько часов или полдня (как мы обсудим в главе 8, графики иногда сдвигаются). Но дело не в этом. Грамотный менеджер проекта собирает максимально полезную информацию о положении команды (и о положении мира) и с помощью нее помогает людям выполнить задачи. Именно эти своевременные фрагменты информации, как в нашем примере, превращают посредственные команды в хорошие, а хорошие – в блестящие. Ни одна система мониторинга проекта или ошибок не заменит полностью человеческое общение, потому что социальные сети всегда сильнее (а иногда и быстрее), чем технологические. Такие масштабные задачи, как в*и*дение проекта, списки функций и график, всегда сводятся к множеству небольших задач, которые легче решать, когда команде доступны нужные знания. МП играет критически важную роль в поддержании потока информации в активном и здоровом состоянии.

Однако независимо от масштаба задач, действия и решения менеджеров должны быть обусловлены пользой для всей команды. Может, пройдет неделя или месяц, прежде чем эта выгода станет очевидна, однако хороший менеджер проекта создает позитивное влияние на качество работы и зачастую на качество жизни всех участников процесса. Люди по-другому относятся к своим обязанностям, лучше понимают, что они делают и зачем, и оптимистичнее, чем раньше, смотрят на будущие задачи и проекты. Изменения происходят постепенно – с каждым следующим собранием, решением или обсуждением, пока атмосфера и энергетика проекта не преобразятся.

МЕНЕДЖЕРЫ ПРОЕКТА СОЗДАЮТ УНИКАЛЬНУЮ ЦЕННОСТЬ

Хорошие лидеры пользуются уважением программистов, тестировщиков, проектировщиков, маркетологов и специалистов по документации. МП должен вырабатывать стратегию, ясно мыслить, проявлять лидерские качества, которые позитивно влияют на команду. Зачастую это предполагает поиск обходных путей и оптимизацию ежедневной работы, а также поощрений в нужное время и нужным способом. МП не обязаны быть сверхлюдьми или даже отличаться особым интеллектом (как я обнаружил на собственном опыте). Им просто нужно понимать преимущества своего положения и использовать их во благо команды.

Есть один простой неоспоримый факт: МП и лидеры уделяют больше времени каждому члену команды, чем кто-либо другой. Они чаще бывают на собраниях, заглядывают в офисы сотрудников и больше остальных руководителей общаются с каждым отдельным работником. Они могут как принимать, так и влиять на большее число решений, чем кто-либо другой в организации. Удовлетворение, разочарование, мотивированность или подавленность лидера так или иначе обязательно отразятся на всех, с кем он общается. Все, что он привносит в проект – как хорошее, так и плохое – заражает остальных.

Итак, если менеджер сосредоточен, вовлечен, воодушевлен и способен добиться успеха, велика вероятность, что другие члены команды возьмут с него пример. Многие менеджеры обладают примерно одинаковой властью и полномочиями и в большинстве случаев могут применять их с немалой пользой. Иными словами, если вообще возможно культивировать отношение и идеи, которые я описал в этой главе, нет людей, более подходящих для этой роли, чем лидер и менеджер. Я не говорю, что МП должен быть харизматичным героем, который одним взмахом руки ведет армии программистов в битву (раздел «Комплекс героя» в главе 11). Напротив, ему просто как можно чаще нужно проявлять искренний интерес и помогать членам команды выполнять поставленные задачи.

В конце концов, если никто не пострадал (кроме, возможно, конкурентов) и вы правильно организовали работу команды, остальное неважно, главное – выполнить то, что нужно выполнить. Не имеет значения, сколько идей предложили вы, а сколько кто-то другой, если результат в итоге положительный. В управлении проектами используются все средства, необходимых для роста вероятности и темпов получения позитивных результатов. Полезный ежедневный принцип, который я применял, звучит так: «Добивайтесь хороших результатов!» Люди видели меня в коридорах компании или в офисе программистов у доски с графиками и таблицами и спрашивали: «Привет, Скотт, чем занят?» Я улыбался и отвечал: «Добиваюсь хороших результатов». Это стало доминирующим методом работы день за днем и хорошо влияло на команду.

Теперь мы перейдем к предметным главам, и я надеюсь, вы тоже почувствуете, как материализуется мой подход и основные идеи из этой вводной главы.

Резюме

Каждая глава книги сопровождается кратким резюме с обозначением ключевых моментов, которые облегчат вам запоминание.

• Управление проектами встречается повсюду и существует с незапамятных времен.

• С «сознанием новичка» перед вами откроется гораздо больше возможностей для обучения.

• Управление проектами может быть должностью, функцией или работой (советы из этой книги применимы ко всем вариантам).

• Программный менеджмент – это четко очерченная роль менеджера проекта в Microsoft, которая опирается на принцип матричной организации.

• Лидерство и менеджмент требуют понимания (в частности интуитивного) нескольких общих парадоксов. Среди них: эгоизм и альтруизм, авторитарность и делегирование, отвага и страх.

• Берегитесь претенциозности и чрезмерной зацикленности на своих менеджерских обязанностях. Процесс должен поддерживать команду, а не наоборот.

• Если вы увлеченный менеджер, найдите способ извлечь преимущества из вашего уникального взгляда на команду и проект.

Упражнения

А. Обратитесь к другу, который работает или учится в другой области. Как он управляет своими проектами? Есть конкретная должность менеджера проекта или эти обязанности распределяются среди разных людей?

Б. Если успешный проектный менеджмент требует сбалансированного подхода, как МП может убедиться, что не заходит слишком далеко в одном либо другом направлении? Как МП может заручиться помощью коллег, чтобы поддержать баланс?

В. Придумайте причину и закатите вечеринку. (Вы пережили первую главу, разве это не достаточная причина?) Когда пройдет похмелье, подумайте: чем вечеринка отличается от проекта? Сравните задачи, проблемы и вознаграждение организатора вечеринок и МП. В чем отличия и в чем сходство?

Г. Вспомните какой-нибудь провальный проект. Чему вы научились и как именно? Перечислите допущенные ошибки. Что можно изменить в следующий раз, чтобы не наступать на те же грабли? Записав это, вы сможете тщательнее обдумать все вопросы и извлечь максимум знаний из своего опыта.

Д. Можете ли вы назвать работу, не требующую управления проектами? Если да, то как ее организовывают и планируют? Какие ограничения создает отсутствие организации? Какие возможности оно дает?

Е. Можете ли вы сами создавать возможности для проявления лидерских способностей или эти ситуации возникают по причинам, не подвластным вашему контролю? Если бы вы хотели увеличить количество возможностей проявить лидерские способности, что бы вы сделали?

Ж. Представьте команду, членов которой вознаграждают исключительно за успешное соблюдение правил и выполнение процессов вместо достижения целей. Как изменится качество работы? Какой будет роль МП? Что это говорит о потенциальных опасностях, которые могут создать МП?

З. Менеджеры среднего звена и их руководители чрезмерно вовлекаются в работу и создают ненужные процессы, потому что занимают среднее положение в организации. Как грамотному менеджеру среднего звена избежать искушений увлечься микроменеджментом и придумать слишком много правил?

Часть первая. Планирование

Глава вторая. Вся правда о графиках работы

* * *

Люди склонны опаздывать. Пусть даже всего на несколько минут или всего несколько раз в неделю, но они отстают от графика. (Отрицание – еще одна потрясающая черта человеческой натуры, поэтому я пойму, если вы считаете, что это не про вас.) Старшеклассники опаздывают на занятия, взрослые – на рабочие встречи. Даже друзья приезжают в бар на десять минут позже. Мы считаем, что «прийти вовремя» – это значит прийти не в определенный момент, а в определенный интервал времени. И для одних этот интервал шире, чем для других. Администраторы ресторанов – любопытный пример. Например, они заверяют, что столик скоро освободится[19], однако на деле это не так. Нам приходится ждать на телефоне или в приемной врача, накапливается богатый опыт разочарований и появляется циничное отношение к графикам как к таковым.

Неудивительно, что столько проектов не укладывается в сроки. Большинство из нас берется за составление графика проекта с отрицательным опытом выполнения или получения результата вовремя. Мы склонны опираться на необоснованные предположения, прогнозировать результаты, рассчитывая на оптимальные условия, и с учетом своего опыта не доверяем графикам, которые видим или создаем. Почему мы так делаем, как это влияет на график проекта и как избежать этих проблем – вот темы второй главы.

[19] Однажды, когда мы с друзьями зашли пообедать в Pizzeria Uno в Питтсбурге, нам сказали, что столик будет готов через десять минут. Ровно через десять минут мой друг Чед МакДаниэль поинтересовался насчет обещанной готовности. Администратор снова ответила, что столик будет готов через десять минут. Чед спросил: «Это те же десять минут или другие?» Шутку она не оценила.

Однако прежде чем пытаться составить более грамотные графики, нужно понять, какие проблемы они решают. Если они настолько ненадежны, зачем вообще тратить на них время? Графики выполняют несколько задач – и лишь немногие из них связаны с прогнозом времени.

Три задачи графика

Любые графики, идет ли речь о подготовке вечеринки или обновлении сайта, служат трем целям. Первая – наметить, когда что будет сделано. График – это своеобразный контракт с каждым участником процесса, подтверждающий, что он выполнит свои задачи за определенный период времени. Как правило, это первое, что вспоминают в связи с графиком проекта. Графики зачастую направлены за рамки проектной команды, на внешние цели, а не на внутренние. Их используют, чтобы заключить сделку или согласовать сроки с клиентом. Последний часто платит не только за услугу, но и за своевременность ее оказания (например, UPS или FedEx). Чтобы дать клиентам и партнерам возможность строить планы, опираясь на конкретный проект, следует обговорить, когда произойдут те или иные вещи.

Вторая задача графика в том, чтобы каждый сотрудник рассматривал свои усилия в рамках общего плана и постарался сделать так, чтобы его вклад сочетался с вкладом других. Пока не будет чернового варианта графика с конкретными датами и сроками, маловероятно, что команда заметит взаимосвязи и зависимости. Без графика все будут заниматься собственными задачами и не поймут, как их работа влияет на других.

Только когда все детали будут записаны с именами людей рядом с каждой задачей, можно сделать реальные расчеты и проанализировать допущения. Это касается даже небольших команд или работников-индивидуалов. График обладает психологической силой, так как публично демонстрирует все обязательства. Сложно забыть или проигнорировать то, что вывешено на доске в коридоре и напоминает всей команде о необходимости выполнения намеченного. Когда есть предварительный график, менеджеры могут поднять вопрос о том, насколько реально выполнить некоторые задачи, и сравнить требования к проекту с имеющимися возможностями.

Этот психологический сдвиг называется функцией принуждения. *Функция принуждения* – это все, что (при реализации) естественным образом вынуждает изменить мировоззрение, отношение или поведение. Итак, графики – важные вынуждающие функции для проектов. При грамотном применении они обязуют всех тщательно обдумать работу, которую нужно сделать. Эта функция – критически важный шаг к реализации потенциала проекта. Даже если та или иная работа сорвется или займет в два раза больше времени, обязательства и связи, выстроенные в результате предварительного графика, можно соблюсти. Итак, вторая задача графика способна полностью сохранить свою ценность, даже если сам график окажется неточным. К примеру, если проект сильно отстает от сроков, наличие графика все равно позволит довести его до конца.

Третья задача графика – дать инструмент для мониторинга и разделения работы на небольшие части. Если разбить работу на задачи, которые занимают день или два, людям легче понять, что нужно делать. Представьте, что строителям дома дали только одно указание: построить дом за 120 дней. С такой слабой детализацией всем, включая прораба, будет сложно понять ход работы. Но если прораб сможет составить еженедельный график работы, то остальные поймут, какие задачи и когда необходимо выполнить, в чем заключаются приоритеты команды, и смогут задать значимые вопросы, уточнить те или иные допущения. С точки зрения МП, хороший график рисует четкую картину проекта, уже на раннем этапе отмечает трудные места, а также повышает вероятность того, что нужный результат будет достигнут.

Чем масштабнее и сложнее проект, тем важнее график. В крупных проектах больше зависимостей между людьми, а решения и сроки сильнее влияют друг на друга. Проблемы в работе

намного проще распознать членам небольшой команды. Когда у последней срывается график, это плохо, однако если опоздание составляет полдня, то понадобится еще полдня и дополнительные усилия всего трех человек, чтобы восстановиться. Кто-то останется в офисе допоздна или, если нужно, вся команда может собраться и помочь наверстать упущенное. На крупных проектах, с десятками или сотнями участников и компонентов, отставание на один день может быстро перерасти в серьезную проблему и вызвать сбой в самых неожиданных областях. Зачастую ни о каком восстановлении речь уже не идет. Но и в большой, и в маленькой команде графики дают менеджерам возможность задавать вопросы, вносить коррективы и помогать команде выявлять проблемы и реагировать на них по мере их появления.

С учетом этих трех задач сразу видно, что даже идеальный график не решает всех проблем проектов. График не исправит плохого проектирования или некорректной разработки и не защитит проект от слабого менеджмента, нечетких целей и неэффективной коммуникации. Несмотря на то что составление хорошего графика тоже требует немалого времени, нельзя забывать, что это всего лишь список слов и цифр. Кто-то должен использовать их как инструмент управления и стимулирования проекта. Так что пора заняться терминологией и изучить методологию программного обеспечения – тяжелую артиллерию проект-менеджмента.

Палочка-выручалочка и методология

Существует множество систем планирования и управления разработкой программного продукта. Их часто называют *методологиями*, то есть сводом практик, нацеленных на достижение определенного результата. Самые распространенные включают в себя каскадную модель, спиральную модель, модель быстрой разработки приложений, экстремальное программирование и разработку по функциональным элементам. Все эти методы направлены на решение схожих организационных или проектных проблем. У каждого есть сильные и слабые стороны, и нужны знания и опыт, чтобы решить, что какому проекту подходит.

Однако моя цель в этой главе, да и в книге, не в том, чтобы сравнивать различные методологии. Напротив, я считаю, что нужно овладеть концепциями, на которые они все опираются, чтобы преуспеть с любой из них. Методологию всегда следует адаптировать и корректировать под спецификации каждой команды и проекта, а это возможно, только когда ваши знания намного глубже поверхностных. Итак, если вы прислушаетесь к основным идеям этой главы и книги, вероятность вашей эффективности возрастет, независимо от того, какой методологией вы пользуетесь. Я объясню аспекты некоторых методов, однако если вам нужен подробный анализ, его придется поискать в других источниках[20]. Хотя методы разработки программного продукта важны, это не палочка-выручалочка. Худшее, что можно сделать, – слепо соблюдать правила, которые явно не работают, просто потому, что они указаны в какой-нибудь известной книге или рекомендуются авторитетным экспертом. Зачастую зацикленность на процессе – тревожный признак: она может свидетельствовать о попытке менеджера уклониться от трудностей и ответственности или погрузиться в бюрократические процедуры, которые заменяют настоящие лидерские действия. Более того, одержимость методологией иногда указывает на слабые стороны организации. Как Том ДеМарко пишет в книге «Человеческий фактор»[21]:

> Одержимость методологией – еще один пример иллюзии высоких технологий. Люди убеждены, что самое главное – технология. ‹…› Но какими бы ни были технологические преимущества, они возможны лишь за счет значительных жертв со стороны команды.

[20] Подробное сравнение традиционных методов разработки программного продукта и agile-метода можно найти здесь: Balancing Agility and Discipline: A Guide for the Perplexed, Barry Boehm and Richard Turner (Addison-Wesley, 2003).

[21] ДеМарко Т., Листер Т. Человеческий фактор. Успешные проекты и команды. СПб.: Символ-Плюс, 2011.

Сосредоточившись на методе и процедуре, вместо того чтобы выстраивать процедуры в помощь людям, МП приступают к планированию, ограничивая вклад отдельных сотрудников. Они акцентируют внимание на правилах и их соблюдении, вместо того чтобы учить сотрудников думать, адаптировать, совершенствовать эти правила. Так что любую методологию следует использовать крайне осторожно: нельзя навязывать ее команде[22]. Напротив, методология должна поддерживать, стимулировать команду, помогать ей достичь результата (в главе 10 вы найдете советы на эту тему).

Соблюдение (или несоблюдение) сроков никогда не зависит только от выбранной методологии. Существуют факторы, которые влияют на все проекты, и МП должны понять их до того, как составлять график работы. Однако прежде чем обсудить это, давайте перечислим компоненты графика.

Как выглядят графики

Есть одно базовое правило для графиков – правило третей. Расчет будет приблизительным или, как говорится, на коленке, однако это самый простой способ понять графики. Если у вас большой опыт их составления, вас сейчас точно передернет – я собираюсь максимально упростить весь процесс. И делаю это с целью буквально на пальцах объяснить, что, почему и как может пойти не по плану и что с этим делать.

Итак, разделите имеющееся время на три части – одна для проектирования, одна для разработки и одна для тестирования. В зависимости от подхода эти задачи получат разное название, однако каждый подход предполагает выделение времени для выполнения этих трех задач. В любой день или в любой час вы решаете, что нужно сделать (планирование), делаете это (программирование) и проверяете, анализируете и корректируете сделанное (тестирование).

Согласно правилу третей, один день вы должны уделить написанию кода, один день – планированию и один день – тестированию и исправлению ошибок (рис. 2.1). Что может быть проще! Это удобный способ проверить существующий график или составить новый с нуля. Если общее количество времени не делится на три части, должны быть очевидные причины, почему проект требует неравномерного распределения усилий. Дисбаланс в правиле третей (например, тестированию уделяется на 20 % больше времени, чем внедрению) допустим, если он не случаен.

[22] Подробнее о том, как осмыслить изменения в процессе разработки софтверного продукта и управлять ими, см. Watts S. Humphrey Managing the Software Process (Addison-Wesley Professional, 1989).

Рис. 2.1. Примитивное правило третей в графике проекта

Возьмем, к примеру, гипотетический проект разработки. Если вам дали шесть недель на запуск, то первое, что нужно сделать, – разделить это время примерно на три части и рассчитать, когда работа будет выполнена. Если для ее качественного выполнения времени не хватает, в ваш проект закралась серьезная ошибка. Либо график следует изменить, либо сократить объем работы (или же снизить планку качества). Если урезать время тестирования, то велика вероятность, что время, отложенное на программирование, вы потратите впустую или же такой код будет сложно сопровождать. Правило третей полезно, так как исключает перетягивание каната, свойственное проектам. Чтобы добавить новые функции, нужен не только программист: неизбежны затраты на проектирование и тестирование, которые кому-то придется взять на себя. Обычно проект отстает от графика из-за скрытых или проигнорированных затрат.

ПОЭТАПНАЯ РАЗРАБОТКА (АНТИПРОЕКТ)

Рассмотрим самый простой пример: нет никакого проекта. Вся работа выполняется фрагментарно: поступает запрос или задача, ее оценивают относительно других задач и ставят на ближайшее «окно» графика. Разработчики сайтов и программисты работают именно так. Они редко вкладываются в масштабные задачи. Подобным командам рекомендуются Agile-методы (скоро мы их обсудим) как самая гармоничная система организации работы, потому что они акцентируют внимание на гибкости, простоте и готовности к изменениям. Если вы трудитесь над несколькими небольшими задачами (не проектами) одновременно, вы можете отталкиваться от примеров проектов, которые я привожу в этой книге.

Правило третей применимо к любым ситуациям. Даже если каждый программист работает самостоятельно над небольшими задачами, он, вероятно, тратит примерно треть общего времени на осмысление того, что нужно сделать, еще треть на саму работу и еще треть на проверку корректности результатов. Он может перескакивать с одной задачи на другую каждые несколько минут, но правило третей применимо для общего понимания любой работы и любых масштабов.

РАЗДЕЛЯТЬ И ВЛАСТВОВАТЬ (БОЛЬШИЕ ГРАФИКИ = МНОГО МАЛЕНЬКИХ ГРАФИКОВ)

В большинстве методологий софтверных разработок явно просматривается правило третей в действии. Конкретные цели и подходы для проектирования или реализации решений могут быть разными, однако в целом желаемые результаты одинаковые.

Ситуация усложняется на масштабных и продолжительных проектах, где графики делятся на небольшие отрезки, внутри каждого из которых выделяются фазы проектирования, реализации и тестирования. Экстремальное программирование называет эти части итерациями; спиральная модель – фазами; а некоторые организации – контрольными точками. Предполагается, что эти отрезки времени занимают всего несколько недель, а спиральная модель уделяет им несколько месяцев, но основная идея неизменна: составить подробные графики по ограниченным периодам времени.

Чем больше изменений ожидается, тем короче каждый этап работы. Это снижает общий риск графика, потому что главный план разделен на контролируемые, посильные части. Перерывы между этапами графика дают возможность внести коррективы и улучшить шансы на то, что на следующем этапе работу удастся срежиссировать более качественно. (Как это сделать, мы обсудим в главе 14.)

Agile и традиционные методы

Экстремальное программирование и другие Agile-подходы предполагают, что будущее всегда изменчиво и непредсказуемо, поэтому они делают ставку на процессы, позволяющие легко изменить направление. Проекты с высокими производственными затратами (например, строительство небоскреба, создание видеоприставки или операционной системы) идут другим путем и вкладывают значительные средства в планирование и проектирование. Это разумно, однако каждый должен воплотить решения, принятые на этапе планирования, и не забывать, что любые изменения связаны с непомерно высокими затратами.

Большинство проектов по разработке находятся примерно посередине. Есть предварительное планирование, однако с учетом изменчивости требований клиентов работа делится на этапы со своими сроками проектирования, реализации и проверки качества. Если возникнет новая проблема, ее можно рассмотреть на текущем этапе работы или отложить в «корзину» и уделить ей внимание уже на следующей фазе.

В большинстве проектов время изначального планирования используется, чтобы получить необходимую информацию от клиентов и бизнес-аналитиков и определить, сколько этапов будет в проекте и чему их посвятить (рис. 2.2): выделить больше времени на проектирование или тестирование, разбить каждый на две небольшие фазы (в соответствии с Agile) или же объединить (монолитный подход). Однако во всех случаях следует предусмотреть время на анализ изменений. Сюда входит решение проблем, которые возникли на предыдущем этапе и им не смогли уделить должного внимания.

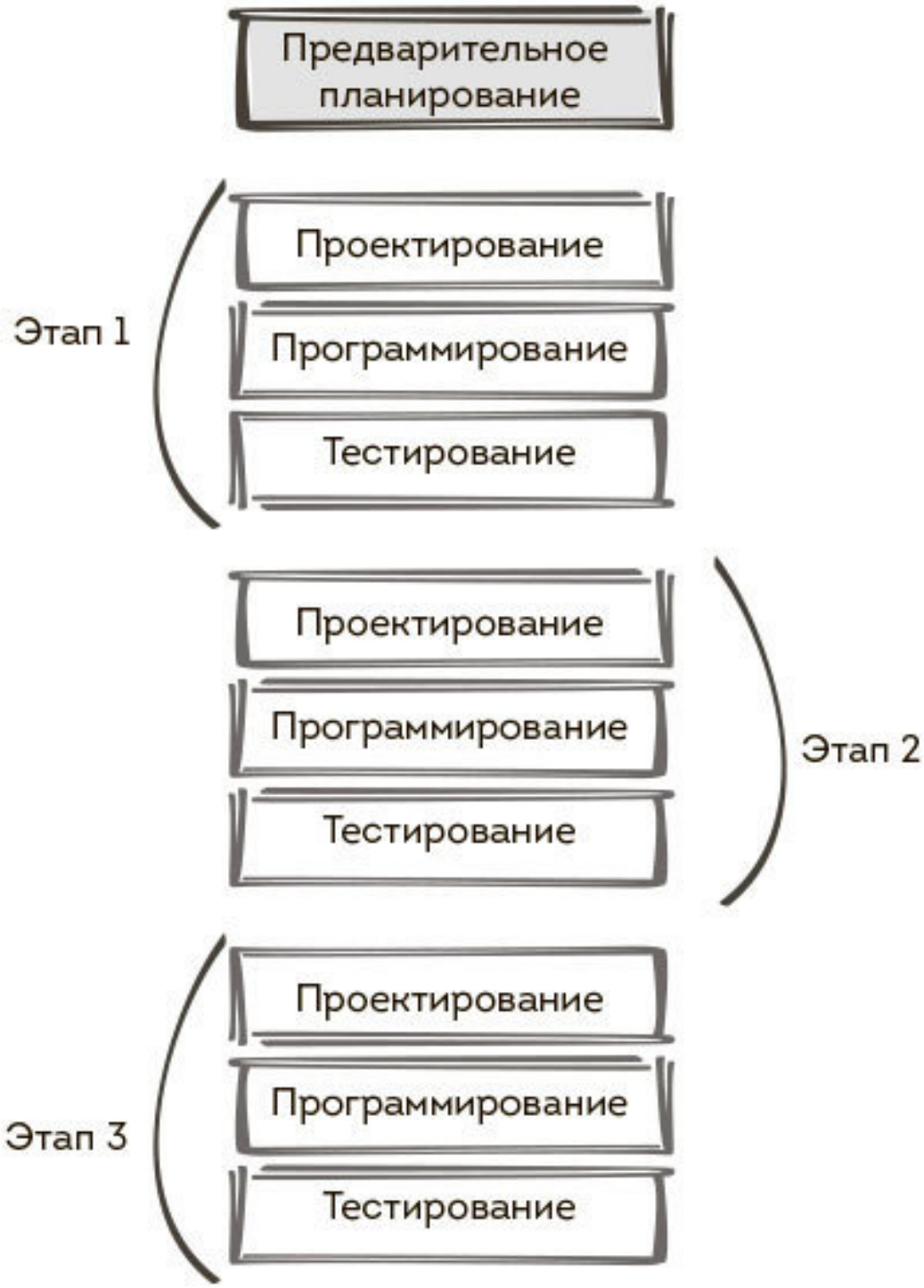

Рис. 2.2. Масштабный проект представляет собой последовательность небольших проектов

Вот и все по поводу методологии планирования. В главе 14 и главе 15 мы поговорим о том, как управлять проектом, однако сделаем акцент на функциях менеджера и лидера, а не на подробном применении конкретной методологии. Если вы прочитали последние два параграфа (даже если вы не согласны с моим мнением), то советы из главы 14 и главы 15 будут актуальны и полезны для вас независимо от того, как вы организуете или планируете свой проект.

В любом случае я извиняюсь перед всеми ветеранами разработки, которые потеряли сознание или почувствовали приступ тошноты во время чтения этого раздела. Он закончился,

и я обещаю, что этот облегченный и упрощенный подход к графикам – практически все, что нужно для понимания изложенных ниже концепций.

Почему графики срываются

Графики проекта – извечный козел отпущений за все, что может пойти не так. Если кто-то ошибется с расчетом или прогнозом, не выполнит требования, даже если кого-то собьет автобус, – во всем виноват график (и тот, кто за него отвечает). Если электричество отключат во всей стране на десять дней или лучшие программисты команды заразятся чумой, кто-то обязательно скажет: «Вот видите, я же говорил, что график сорвется», – и ткнет пальцем в того, кто его составлял. Это абсолютно несправедливо, но происходит постоянно. Несмотря на всю свою ненависть к графикам, люди предъявляют к ним завышенные требования. Даже лучшие графики в мире, с лучшими умами и лучшими инструментами, – это всего лишь попытка прогнозировать будущее, а человеку она редко удается.

Если команда приступает к проекту, осознавая вероятные причины, по которым график может сорваться, и предпринимает определенные действия, чтобы свести эти риски к минимуму, график станет гораздо более полезным и точным инструментом процесса разработки.

СТРЕЛЬБА ВСЛЕПУЮ С ОГРОМНОГО РАССТОЯНИЯ

Если график составляется на этапе изначального планирования, сотни решений, которые могут повлиять на него, еще не приняты. Будут трудности и вопросы, которые никто не способен предугадать, и на раннем этапе ориентировочного плана невозможно их учесть. Пока люди не поймут требования и не приступят к проектировке, у менеджера проекта слишком мало информации для реалистичных прогнозов. Однако зачастую черновой график составляют с выдуманными цифрами и совершенно дикими прогнозами, и это «соломенное чучело» вручают команде под видом надежного плана проекта. Его авторы часто становятся жертвами стремления к детализации (вместо того чтобы бросить все силы на достоверность): впечатляющий график с конкретными датами и сроками (детализация) далеко не всегда отражает реальное положение дел (достоверность). Детализация – довольно простая задача, в отличие от достоверности.

Однако все проекты и графики действительно должны с чего-то начинаться. Метод «пальцем в небо» тоже можно использовать – чтобы вдохновить команду и установить определенные границы. Так можно начать процесс анализа и исследования, чтобы конкретизировать график, поднять важные вопросы и ответить на них. Но если в основу графика легли непроверенные и неизученные залихватские предположения, причем без дальнейшей корректировки, вы сильно рискуете. Факты говорят о том, что любому человеку сложно рассчитать необходимое время на раннем этапе проекта.

Барри Боэм в своем эссе 1988 года на тему разработки ПО[23] пришел к выводу, что ошибки графиков растут пропорционально тому, насколько рано сделаны предположения (рис. 2.3). Если общий расчет графика осуществлен рано, он может попасть мимо цели аж на 400 %, причем в обоих направлениях (подозреваю, что ошибки в графике всегда играют против нас, отнимая больше времени, чем мы ожидали). На этапе проектирования, когда многие решения становятся очевидными, диапазон отклонений от графика немного сужается. Только когда проект перейдет в стадию реализации, расчеты становятся обоснованными и реалистичными, но даже тогда остается 20 %-ная вероятность того, что график недостаточно точен.

[23] “Understanding and Controlling Software Costs,” IEEE Transactions on Software Engineering, vol. 14, no. 10, October 1988, pp.1462–77; а также Barry Boehm Software Engineering Economics (Prentice Hall, 1991).

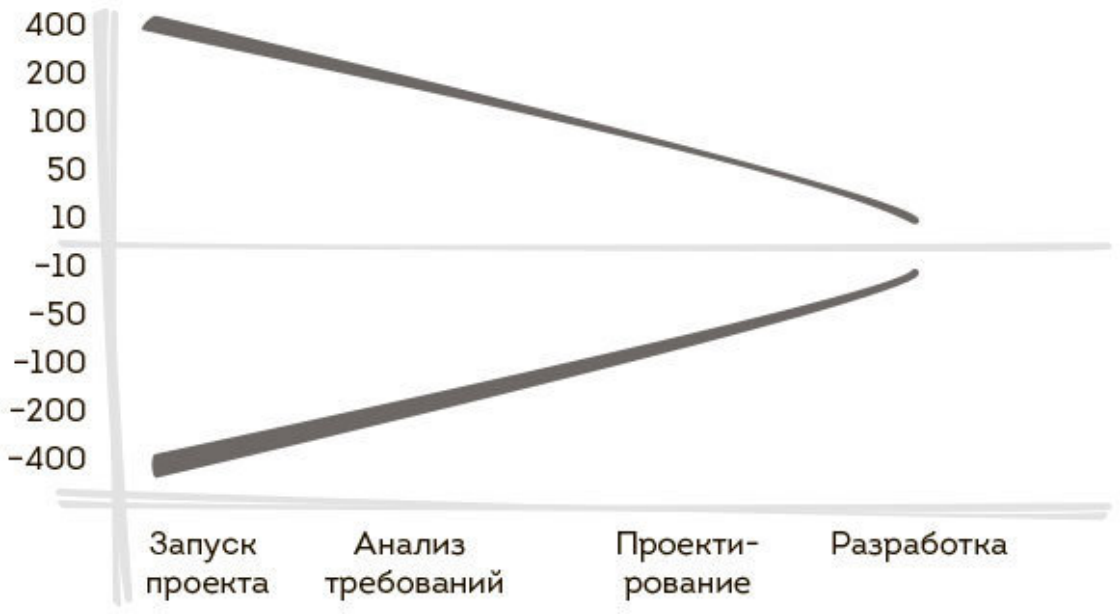

Рис. 2.3. Диапазон возможных отклонений от сроков проекта (адаптировано из Software Engineering Economics Боэма)

МП должны понимать, что со временем расчет графика становится точнее и обоснованнее. Графики требуют, чтобы на них обращали внимание по ходу работы и вносили корректировки по мере развития проекта.

ГРАФИК – ЭТО ВЕРОЯТНОСТЬ

Когда я работал над своими первыми масштабными проектами после окончания колледжа (Windows и Internet Explorer), кто-то гораздо более авторитетный, чем я, присылал сверху график моей команде. Поскольку я был неопытным новичком и не мог активно участвовать в процессе, мне и небольшому количеству программистов и тестировщиков оставалось только следовать этому «генплану».

Мы бурно обсуждали и согласовывали его отличия от графика, который составляли опираясь на конкретные задачи[24]. «Генплан» спускали сверху, и он был великолепно «причесанным», с аккуратными колонками, датами и цифрами. Словно некий артефакт, выкраденный из будущего.

Несмотря на наш цинизм, мы все же следовали «генплану». Хотя его происхождение оставалось туманным, мы доверяли тимлидам, а к тому же мы были слишком заняты своей работой. (Вообще-то они вполне могли объяснить нам эти графики, но мы были слишком заняты и слишком доверяли им, чтобы тратить на это время.)

Позже, начав составлять графики, я осознал одну истину. Их трудно назвать подарками из будущего. Нет никакой волшебной формулы или науки для идеальных графиков. Несмотря на неопытность, я понимал, что их составление всегда затрагивает множество разных аспектов текущего и будущего положения проекта. График – это просто прогноз сроков. Неважно, насколько точно он составлен или насколько убедительным кажется, – это всего лишь свод-

[24] Графики, основанные на задачах программистов, принято называть графиками «снизу вверх». Графики, основанные на решении менеджмента, называются графиками «сверху вниз». Как правило, отличия между ними обговариваются и согласуются.

ная запись множества мелких расчетов, причем каждый из них неизбежно подвержен самым разным непредвиденным упущениям и проблемам. Хорошие графики создают только лидер или команда, которые стремятся к точным суждениям по многим аспектам программной разработки. Для эксперта в одной узкой области производства это будет не под силу.

Итак, если все члены команды согласны, что график – это ряд предположений, то проблемы не в нем самом, а в его применении. Если график показывают на встрече команды или отправляют всем по электронной почте, возникает вопрос: насколько реальны указанные сроки? Если, например, не указаны пять самых вероятных рисков и прогнозы по ним, и тот, кто составил график, не может объяснить свои предположения, следует признать, что график возможен, но нереален[25]. Команда должна высказать свое мнение – какие соображения и какую информацию можно добавить или изменить, чтобы он стал более правдоподобным.

График работы вовсе не должен быть идеальным (большое облегчение, конечно, потому что идеальных графиков не существует). Достаточно, если он будет убедительным для команды и лидеров, допускать мониторинг и внесение поправок, а также указывать на вероятность успеха, которая удовлетворяет клиента, бизнес или спонсора проекта.

ДЕЛАТЬ ПРОГНОЗЫ СЛОЖНО

На этапе проектирования (глава 5 и глава 6) одна из задач команды – разбить проект на небольшие части. Эти части, которые нередко называют элементами (единицами) работы или иерархической структурой работ (work breakdown structure, WBS[26]), становятся пунктами общего графика проекта. Элементы работы (скрестим пальцы) оптимально распределяются[27] среди программистов, затем подсчитывается время на создание каждой из них и выстраивается график. Программист должен выделить каждому элементу работы определенное время и на основе этих прогнозов составить график.

Если брать простейшее определение, грамотные расчеты рабочего времени имеют высокую вероятность точности, а неграмотные расчеты – низкую. За такие определения я не жду никаких наград, однако в них есть как минимум одна польза: решения тимлидов определяют стандарты для каждого проекта. Приходится активно анализировать прогнозы, понукать, побуждать и стоять над душой, чтобы добиться нужного уровня. Думаю, к составлению прогнозов вполне разумно привлечь тестировщиков или команду контроля качества, чтобы они участвовали в решениях по проектированию, задавали вопросы и высказывали свое мнение. Как минимум это поможет им самим составить надежный график тестирования, который далеко не всегда согласуется с прогнозами программистов. Зачастую контроль качества лучше всего проясняет пробелы проектирования и возможные изъяны, которые остальные упускают из вида.

Мир основан на прогнозах

Прогнозы – задача непростая, поскольку мало кому нравится рассчитывать сложные вещи, да еще нести за это ответственность. Намного приятнее хвастаться своими способностями («Эта книга, фильм или сайт отстой: я бы сделал намноооооого лучше!»), но когда тре-

[25] Любой график – лишь один из многих возможных вариантов. В зависимости от того, какие непредвиденные факторы (ошибка проектирования, политическая революция, нападение пришельцев и т. д.) в нем учитываются, сроки могут быть разными. Без учета вероятных ошибок график не может быть надежным: его автор не проявил достаточно креатива или скептицизма.

[26] Многие учебники и руководства описывают, как выстроить структуру разделенной работы. Я коснусь этой темы в главе 14, однако если вы хотите получить более подробную информацию, начните с http://en.wikipedia.org/wiki/Work_breakdown_structure или Total Project Control, Stephen Devaux (Wiley, 1999).

[27] В книге Кента Бека Extreme Programming Explained (Addison-Wesley, 1999) (Бек К. Экстремальное программирование. Разработка через тестирование. СПб.: Питер, 2017) предлагается модель распределения работы для программистов, где они сами выбирают рабочие единицы. В идеале это решение – компромисс между благом проекта и благом членов команды.

буется взять дело в свои руки и выдать результат – поставить подпись на договоре с подробным перечнем наших обязанностей, – все меняется. Как известно, что бы мы ни пообещали сделать, в момент выполнения обещания все покажется невозможным и нежелательным. Ситуация грозит оказаться намного сложнее, чем мы думали. Программисты похожи на всех остальных людей и вполне оправданно боятся делать какие-либо прогнозы. Если они скажут, что задачу можно выполнить за определенное время, то рискуют сесть в лужу.

Как показывает мой опыт, даже те, кто понимает процесс прогнозирования и верит в него, не любят этим заниматься. Отчасти из-за нестыковки предположений («Как же это сделать, если у меня так мало информации?») с временны*ми* рамками («Скажите точно, сколько часов займет эта работа»). Однако никаких поблажек быть не должно: все, кто работает в проектировании и строительстве, решают примерно одинаковые «головоломки», будь то строительство небоскреба, ремонт кухни или запуск космического корабля на другие планеты. Если почитать, как все эти люди делают прогнозы, то вы увидите, что их проблемы и методы не слишком отличаются от сфер разработки или программного обеспечения. Основное отличие в том, сколько времени им дают на расчеты и насколько дисциплинированно они его распределяют (мы обсудим это подробнее в главе 5 и главе 6).

ГРАМОТНЫЕ ПРОГНОЗЫ ЗАВИСЯТ ОТ КАЧЕСТВЕННОГО ПРОЕКТИРОВАНИЯ

Самое важное, что я узнал о грамотных прогнозах, – их можно получить только при качественном проектировании и ТЗ. Хорошие технические расчеты зависят от надежной информации и квалифицированных специалистов. Если спецификация никуда не годится, а программиста просят назвать цифры, опираясь на совершенно непонятные каракули на доске, все должны понимать, что получат весьма расплывчатые подсчеты. Проще говоря, прогнозы касаются всех, и команда должна сделать все возможное, чтобы помочь проектировщикам сделать их верными. Если разработка прогнозов кажется тяжелой повинностью или если в этом процессе не участвуют тимлиды, не ожидайте ничего надежного и правдоподобного.

Если лидеры допускают недостоверные прогнозы в графике и не возражают против высоких рисков, то это не проблема. В небольших, коротких проектах приблизительные прогнозы – как раз то что нужно. Требования часто меняются, а бизнес умеет быть гибким. Нет ничего плохого в некачественных прогнозах, если никто не путает их с качественными.

Я обнаружил один полезный метод: когда программист ворчал, что от него требуют прогнозов по проекту, я спрашивал: «На какие вопросы я могу ответить, чтобы у тебя было больше уверенности относительно сроков?» Направляя разговор к большей конкретике, я давал ему возможность высказать все испытываемые им страхи и недовольства и помогал решить проблему. Конечно, мне приходилось часть работы брать на себя и доказывать, что он выполняет далеко не все свои обязанности, но в любом случае мы обсуждали, как составить надежные прогнозы.

Перечислим еще несколько методов.

• **Определить базовые показатели доверия прогнозам.** Догадка = 40 % доверия. Обоснованный расчет = 70 %. Детальный и тщательный анализ = 90 %. Лидеры команды должны согласовать, какая точность им нужна, сколько времени они дают программистам на ее расчеты и как справиться с рисками, если прогнозы не оправдаются. Не зацикливайтесь на цифрах: используйте их просто для того, чтобы четко обозначить качество прогнозов. Доверие на уровне 90 % означает, что прогнозы оправдываются в 9 случаях из 10. Если вы решили попросить команду улучшить качество прогнозов, то нужно дать ей больше времени.

• **Ведущие разработчики должны установить планку качественных прогнозов, задавая правильные вопросы и выбирая грамотные подходы, на которые команда**

сможет равняться. Делайте все необходимое, чтобы на корню пресечь желание давать фальшивые прогнозы или отказываться от своих слов (например, «Я ни за что не отвечаю», «Это всего лишь предположение» и т. д.). Выясните, какая информация нужна людям для грамотного прогноза, и предоставьте время, соответствующее поставленной цели.

• **Программистам нужно доверять.** Если нейрохирург сказал вам, что операция займет пять часов, будете ли вы настаивать, чтобы он уложился в три? Сомневаюсь. Иногда необходимо поднажать, чтобы люди объективно взглянули на ситуацию – но только в качестве уравновешивающего средства (классический пример: когда программист рассчитывает, что задачи, которые ему не нравятся, займут много времени, а задачи, которые ему нравятся, займут мало времени). При необходимости можно сравнить несколько прогнозов (от двух разных разработчиков) и проверить их достоверность.

• **Прогнозы зависят от того, насколько хорошо программист понимает цели проекта.** Прогнозы строятся на том, как программист интерпретирует спецификации (если они есть), а также цели и задачи проекта. В своей книге «Психология программирования» (The Psychology of Computer Programming) (Dorset House, 1971) Джеральд Вайнберг показывает, как отсутствие понимания основных целей напрямую воздействует на предположения программистов. Даже если технологическая задача предельно ясна, подход программиста к ее решению может значительно измениться в зависимости от общего направления всего проекта.

• **Прогнозы должны опираться на предыдущие результаты работы.** Хорошая привычка для программистов – отслеживать свои прогнозы по проектам. Им следует обсуждать это со своим менеджером, который должен стремиться понять, кто в его команде делает грамотный «мониторинг». Экстремальное программирование использует термин *производительность*, когда речь идет о вероятных результатах работы программистов (или команды) в зависимости от их предыдущих результатов[28].

• **Качество спецификаций и проектирования диктует, насколько точными должны быть прогнозы технических специалистов.** Это обсуждается между менеджерами проекта и программистами. Чем выше требования по прогнозам, тем выше должно быть качество спецификаций. Подробнее о спецификациях мы поговорим в главе 7.

• **Существуют методы построения грамотных прогнозов.** Самый известный из них – PERT[29], который позволяет минимизировать риски, выводя средний показатель по высоким, средним и низким прогнозам. Метод хорош по двум причинам. Во-первых, это заставляет всех осознать, что прогнозы – лишь предварительный диагноз и что результаты могут быть совершенно разными. Во-вторых, это дает менеджеру проекта возможность регулировать агрессивность или консервативность графиков (больше веса можно приписать низким или высоким прогнозам).

САМЫЕ ЧАСТЫЕ УПУЩЕНИЯ

Хотя надежные прогнозы намного улучшают качество графиков, многие факторы, влияющие на графики, не связаны с самой работой. И получается, что даже если ваши прогнозы по единицам работы идеальны, реальные риски графика связаны с теми факторами, которые нигде не учитываются. Хотя шансы подхватить чуму крайне малы в большинстве стран мира, вероятность того, что главный программист заболеет гриппом или уедет в отпуск, довольно высока. Есть ряд распространенных упущений графика, которые должны знать все менеджеры

[28] См. Kent Beck and Martin Fowler, Planning Extreme Programming (Addison-Wesley, 2002) (Бек К., Фаулер М. Экстремальное программирование. Планирование. СПб.: Питер, 2003), pp. 60–62.

[29] PERT – Program Evaluation and Review Technique, метод оценки и анализа проектов. Стандартная формула: (оптимистический прогноз + (4 × наиболее вероятный прогноз) + пессимистический прогноз) / 6. Однако существует масса вариаций и теорий о том, как лучше рассчитывать взвешенную оценку.

проекта. Обжегшись на одном упущении, вы захотите отслеживать его в будущем. Вот почему проектный менеджмент – и управление графиком в частности – требуют опыта. Есть масса разных вариантов сесть в лужу и только один способ научиться обходить упущения – нести ответственность за последствия.

Предлагаю личный список вопросов, которые помогли мне предусмотреть потенциальные проблемы графиков на ранних этапах. Чаще всего я спрашивал, что пошло не так, уже после завершения проекта и пытался сформулировать вопрос, который можно было бы задать раньше, чтобы избежать проблемы. (Чего не хватило? Что мы не учли? Что бы изменило ситуацию или позволило бы мне внести коррективы?)

• Учтены в графике больничные и отпуска всех участников процесса?

• Заложены в график сезонные праздники или другие периоды года, когда работа заходит в тупик (например, со Дня благодарения до Рождества в США, лето в Европе)? Важные дедлайны крайне тяжело соблюдать в этот период.

• Был ли доступ к графику у членов команды? Просили ли их регулярно отчитываться о проделанной работе (не вызывая при этом раздражения)?

• Кто-нибудь следил за общим графиком ежедневно или еженедельно? Было ли у этого человека достаточно полномочий, чтобы задавать грамотные вопросы и вносить коррективы?

• Проявляла ли команда интерес к графику? Если нет, то почему? Участвовала ли она в построении графика и выборе задач, которые следует выполнить, или график спустили сверху?

• Лидеры команды добавили больше функций или помогли убрать больше функций? Лидеры команды когда-нибудь говорят «нет» на новые задачи и дают разумные указания команде о том, как отвечать на новые (поздние) задачи?

• Членов команды поощряют отказываться от новой работы, которая не вписывается в задачи и в*и*дение проекта?

• Какой процент вероятности был использован в прогнозах – 90 %, 70 %, 50 %? Это было отражено в общем графике? Клиент или вице-президент учитывали это? Обсуждались ли другие предложения, которые заняли бы больше времени, но имели более высокую вероятность?

• Выделено в графике время для его корректировки и дискуссий между лидерами и менеджерами?

• Учитывал ли график сокращение рабочих часов во время праздников? Внесены в график потенциально опасные погодные изменения?

• Были ли достаточно качественными спецификации и план проекта, чтобы разработчики сделали на их основе грамотные прогнозы?

• Обладают ли специалисты, осуществляющие расчет, должной подготовкой и опытом?

Конец ознакомительного фрагмента.