ГЛАВА 9

ЕЩЕ БОЛЕЕ ЭФФЕКТИВНОЕ ВЫПОЛНЕНИЕ ПРОЕКТОВ

В предыдущей главе мы рассмотрели, как организовывать и поддерживать разработчиков при работе по Agile. В этой главе обсуждается, как правильно организовать и поддерживать процесс разработки при работе по Agile.

Большую часть разработки ПО организуют в проекты. В организациях используют самые различные термины для обозначения понятий, связанных с проектом, в том числе «продукт», «программа», «релиз», «цикл релиза», «функция», «поток создания ценности», «рабочий поток» и прочее.

Терминология значительно различается. Некоторые компании считают, что понятие «релиз» — это синоним слова «проект». Другие думают, что понятие «релиз» относится к последовательной разработке, поэтому прекратили его использовать. Третьи определяют понятие «функция» как объем работ, рассчитанный на 3–9 человек и 1–2 года. В этой главе под словом «проект» я буду иметь в виду любые из перечисленных видов работ, то есть работу некоторого количества сотрудников в течение длительного времени.

КЛЮЧЕВОЙ ПРИНЦИП: НЕБОЛЬШИЕ РАЗМЕРЫ ПРОЕКТОВ

За последние 20 лет наиболее известны случаи успешного применения Agile в небольших проектах. Agile первые 10 лет своего существования уделял большое внимание тому, чтобы проекты были небольшими, то есть 5–10 человек в команде (например, 3–9 разработчиков, владелец продукта и скрам-мастер). Акцент на небольшом размере проектов очень важен, потому что с небольшими проектами успешно справиться легче, нежели с крупными, как показано на рис. 9.1.

Каперс Джонс уже 20 лет постулирует, что мелкие проекты чаще завершаются успешно, чем крупные [Джонс, 1991; 2012]. Я обобщил большинство исследований о зависимости успешности проектов от размера в своих книгах «Совершенный

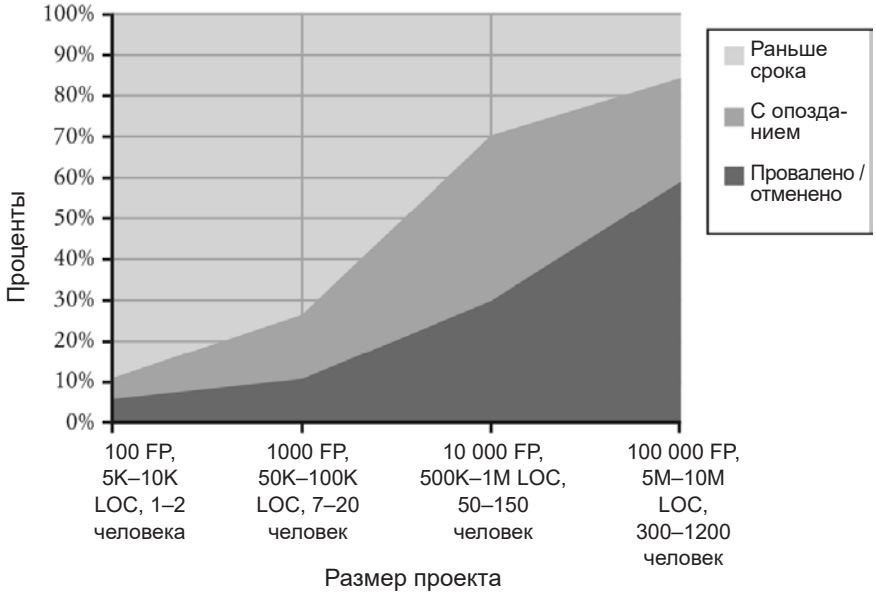


Рис. 9.1. Чем больше проект, тем выше вероятность сорвать сроки, не уложиться в бюджет и выше риск провала (Джонс, 2012). Проект можно измерять в единицах функций (FP), а также в строках кода (LOC). Сравнения размеров, приводимые в единицах функций, строках кода или размеров команд, приблизительны

код» [Макконнелл, 2004; СПб.: Питер, 2007] и Software Estimation: Demystifying the Black Art [Макконнелл, 2006].

Небольшие проекты благополучнее по нескольким причинам. Крупные проекты требуют вовлечения большего количества специалистов, и число взаимосвязей между людьми в командах и между самими командами увеличивается в нелинейной прогрессии. А чем более сложными становятся взаимоотношения, тем больше ошибок совершается при взаимодействии. Ошибки взаимодействия приводят к ошибкам в требованиях, проектировании, написании кода — в общем, к другим ошибкам.

Следовательно, чем крупнее становится проект, тем больше допускается ошибок (рис. 9.2). Это говорит не просто о том, что общее количество ошибок растет, а о том, что в крупных проектах несоизмеримо больше ошибок!

Чем выше частота появления ошибок, тем ниже эффективность стратегий обнаружения недочетов. Это означает, что количество недочетов в готовом продукте непропорционально возрастает.

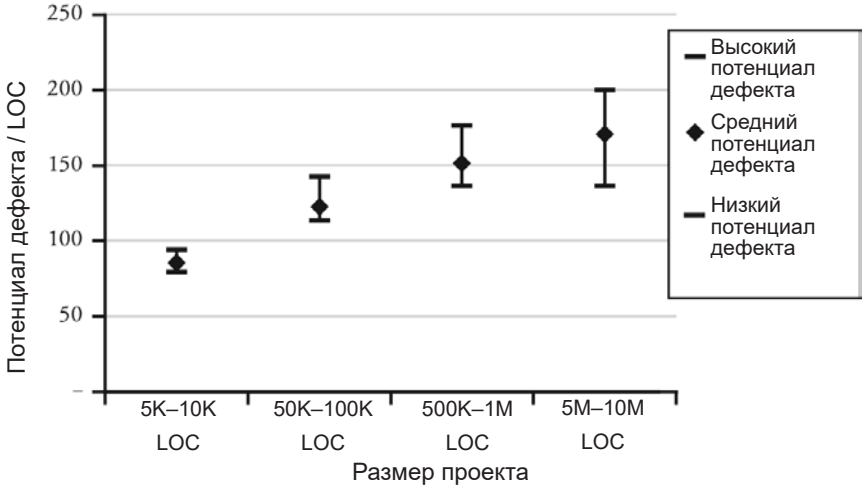


Рис. 9.2. Чем крупнее проект, тем больше частота появления ошибок (потенциал дефекта). Адаптация исследований [Джонс, 2012]

Требуется также больше усилий для устранения ошибок. Следовательно, как показано на рис. 9.3, более мелкие проекты отличаются большей производительностью на человека, но чем больше размер проекта, тем сильнее падает производительность. Это явление известно как «издержки масштаба».

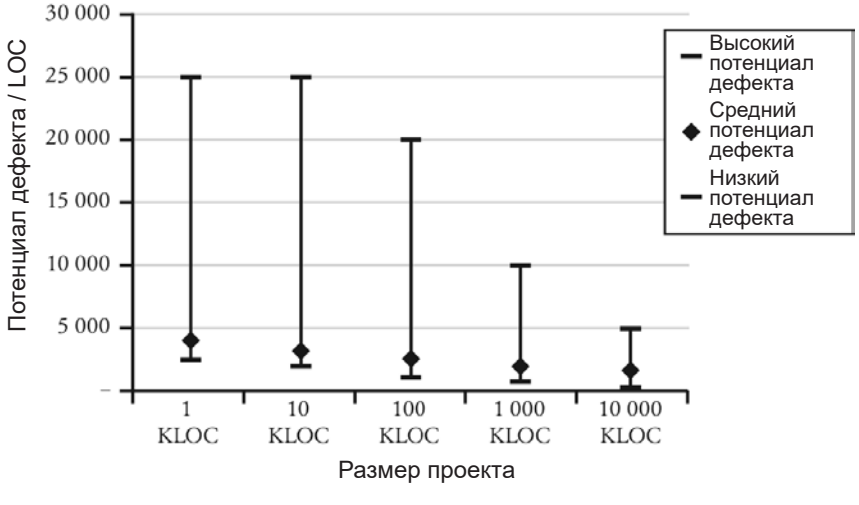


Рис. 9.3. Чем крупнее проект, тем ниже производительность на человека. Адаптация исследований [Макконнелл, 2006]

Эту обратную зависимость между размером и производительностью тщательно исследовали и проверяли на протяжении более 40 лет. Фред Брукс рассуждал на тему издержек масштаба при разработке ПО в первом издании своей книги «Мифический человеко-месяц» [Брукс, 1975; СПб.: Питер, 2021]. Работа Ларри Патнэма по оценке издержек при создании ПО подтвердила наблюдения Брукса [Патнэм, 1992]. Исследование модели издержек разработки (Cocomo) эмпирически подтвердило существование издержки масштаба дважды — в оригинальном исследовании конца 1970-х годов и в обновленном исследовании конца 1990-х [Боэм, 1981; 2000].

Вывод таков: чтобы максимально увеличить вероятность успешного выполнения проекта по Agile, старайтесь сохранять небольшие масштабы проекта (и команды).

Конечно, невозможно сделать так, чтобы каждый проект был мелким. В главе 10 «Еще более эффективное выполнение крупных проектов» изложены подходы к крупным проектам, в том числе предложения о том, как их уменьшить.

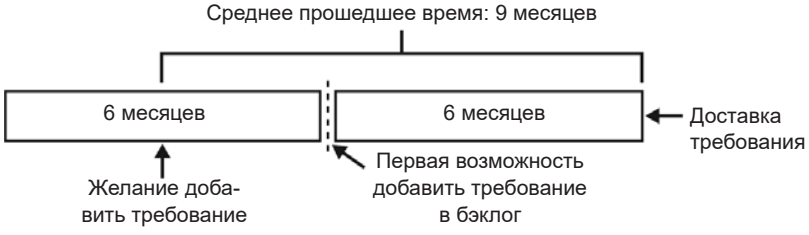
КЛЮЧЕВОЙ ПРИНЦИП: КОРОТКИЕ СПРИНТЫ

Естественный вывод из того, что предпочтительны небольшие размеры проекта, — такой, что спринты не должны длиться долго. Можно подумать, что небольшой проект уже сам по себе залог успеха. Но короткие спринты в 1–3 недели способствуют всестороннему успешному ведению проектов. Это описано в следующих нескольких разделах.

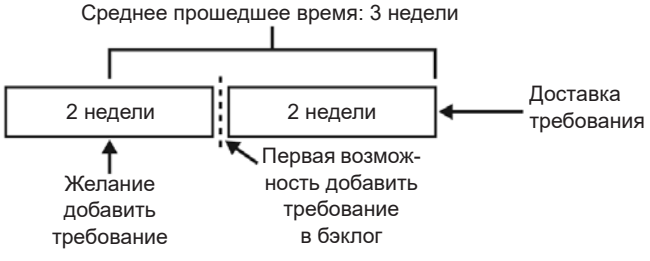
Короткие спринты снижают количество промежуточных требований и улучшают реагирование на новые требования

В Скраме новые требования можно добавлять между спринтами. Но когда спринт начался, до следующего спринта требования добавлять уже нельзя. Это разумно, если спринт длится 1–3 недели.

Если циклы разработки длятся дольше, то стейкхолдеры по ходу работы все настойчивее просят добавить требования, поэтому просьбы отложить добавление требований уже не так обоснованны. Если цикл при последовательной модели разработки длится полгода, то попросить стейкхолдера отложить реализацию нового требования до следующего цикла означает, что работа над требованием начнется только в следующем цикле — то есть в начале цикла это требование только добавят, а потом нужно ждать доставки продукта в конце этого цикла. В среднем это занимает 1,5 цикла, то есть 9 месяцев.



В противоположность этому обычные для Скрама спринты длятся две недели. Это означает, что стейкхолдеру, который желает добавить новое требование, придется подождать в среднем три недели.



Просить стейкхолдера подождать 9 месяцев для реализации требования зачастую неуместно. А три недели — почти всегда уместно. Это значит, что скрам-команда спокойно работает, не боясь появления новых требований посреди спринта.

Короткие спринты обеспечивают большую оперативность при работе с клиентами и заинтересованными сторонами

Каждый спринт представляет собой новую возможность для демонстрации рабочего ПО, проверки требований и проработки обратной связи со стейкхолдерами. При стандартных двухнедельных спринтах команды 26 раз в год дают себе возможность быть оперативнее! При трехмесячном же цикле разработке такая возможность представляется лишь четыре раза в году. Пятнадцать лет назад проект, рассчитанный на три месяца, считался коротким. Сегодня такой график означает, что вы не сможете оперативно реагировать на обратную связь от стейкхолдеров, клиентов и рынка.

Короткие спринты укрепляют доверие стейкхолдеров

Поскольку команды чаще демонстрируют результаты, ход работ становится прозрачнее, поэтому для стейкхолдеров его продвижение налицо, а это укрепляет доверие между ними и командой.

Короткие спринты способствуют скорости разработки посредством циклов «изучайте и приспосабливайтесь»

Чем выше частота итераций, тем больше у команды возможностей поразмышлять над полученным опытом, сделать выводы и применить их результаты к практическим методам в работе. Для этой области подходит то же самое обоснование, что и для оперативности работы с клиентами: что лучше — позволить своим командам изучать и приспосабливаться 26 раз в год или только четыре? Короткие спринты помогают вашей команде совершенствоваться быстрее.

Короткие спринты помогают сократить время на эксперименты

В контексте «Запутанные» по Cynefin проблемы нужно сначала исследовать, прежде чем получится понять весь объем работ. Такие исследования нужно характеризовать так: «Чтобы получить ответ на тот или иной вопрос, сделайте наименьшее необходимое количество работы». К сожалению, закон Паркинсона гласит, что работа занимает весь доступный объем времени. А до тех пор, пока у команды не разовьется железная дисциплина, решение вопроса, если на него выделен месяц, и займет ровно месяц. Но если есть всего две недели, то чаще всего и решение вопроса занимает две недели.

Короткие спринты позволяют вовремя обнаружить издержки и риски

Короткие спринты дают возможность чаще отслеживать состояние хода работ. По истечении лишь нескольких спринтов при работе над новой задачей выяснится «скорость» команды или темп хода работ. Возможность наблюдения за продвижением работы облегчает прогноз сроков выпуска продукта в релиз. Если работа занимает больше времени, чем изначально планировалось, это станет предельно ясно всего через несколько недель. Небольшая длительность спринтов — мощный инструмент для осознания положения дел. В главе 20 «Еще более эффективная предсказуемость» рассказано об этом более подробно.

Короткие спринты способствуют подотчетности команды

Если команда несет ответственность по доставке рабочего функционала каждые две недели, у нее нет возможности долго работать «в темную». Она показывает результаты своей работы на встречах для обзора спринта, а также отчитывается перед стейкхолдерами каждые две недели. Еще чаще плоды труда видит владелец продукта.

Владелец продукта принимает или отклоняет работу, ход работы хорошо просматривается. Таким образом, команды лучше отчитываются за свою работу.

Короткие спринты способствуют подотчетности

На протяжении поколений команды разработчиков страдали от «звезд», которые запирались на несколько месяцев в темную комнату, и было совершенно неясно, продвигается ли работа. В случае со Скрамом такой проблемы больше нет. Каждый работает в поддержку целей команды в спринте, к тому же существует необходимость каждый день приходить на стендапы и рассказывать о том, какая работа вчера была выполнена, — запереться больше не получится. Либо разработчик начинает сотрудничать, что решает проблему одним способом, либо не выносит условий и уходит из команды, что все равно решает проблему, хотя и другим способом. По своему опыту могу сказать, что любой из исходов благоприятнее, чем в случае, когда кто-то работает без каких-либо отчетов недели или месяцы, а в итоге обнаруживается, что ничего толком не сделано.

Короткие спринты способствуют автоматизации

Поскольку команды часто выполняют сверку, спринты небольшой длительности способствуют автоматизации задач, которые бы в ином случае были бы однообразны и затратны по времени. Автоматизация распространена в том числе при сборке, интеграции, тестировании и статическом анализе кода.

Короткие спринты чаще доставляют чувство удовлетворенности

Команда, которая доставляет работающее программное обеспечение каждые две недели, с завидной частотой ощущает удовлетворенность своей работой, а также чаще располагает возможностью отметить свои достижения. Это вносит свой вклад в чувство профессионализма, которое способствует мотивации.

КОРОТКИЕ СПРИНТЫ. РЕЗЮМЕ

В целом пользу коротких спринтов можно кратко характеризовать так: «Скорость доставки во всех отношениях помогает справиться с объемом работ». Доставка функционала небольшими партиями и короткие каденции приносят огромное количество преимуществ по сравнению с доставкой функционала большими партиями и длинными каденциями.