1 Важная роль код-ревью

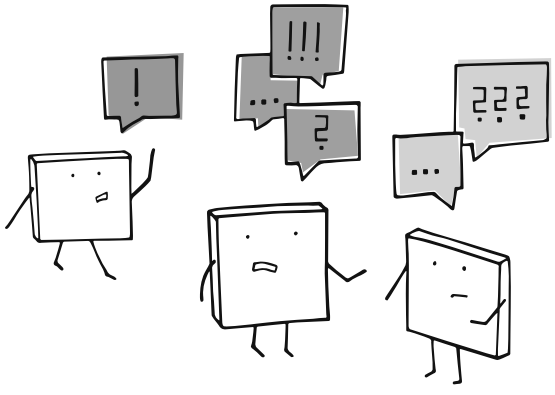
В этой главе

- ✓ Введение в процесс код-ревью
- ✓ Для кого эта книга (и кому она не нужна)
- ✓ Структура книги
- ✓ Преимущества, которые дают обзоры кода
- ✓ Улучшение процесса рецензирования

Майк, разработчик из небольшой команды, закончил работу над новой функцией в пятницу. Он написал абсолютно новый парсер PDF-файлов для выделения сумм из присланных клиентами счетов. И успел он ровно в срок — до начала своего отпуска, который намеревался провести в Мексике. Конечно, не обошлось без костылей и палок, но все *работало*. Довольный собой, Майк, недолго думая, загрузил свой код в демосреду. Эдриенн, Эрика и Джастин — другие члены команды — согласились показать новую фичу генеральному директору, пока Майк будет отдыхать.

В понедельник утром Эдриенн начала тестировать новую программу и с удивлением обнаружила, что вычисления выполняются неправильно. Отладить код не получалось, поскольку в нем невозможно было разобраться. Призванные на помощь Джастин и Эрика спустя какое-то время лишь развели руками. Пока

все трое пытались расшифровать загадочный код, Майк слал им картинки с пляжа — море, коктейли, закуски...



К сожалению, к моменту демонстрации функции генеральному директору разработчики не смогли решить проблему с расчетами. Переписать демонстрационный код также не удалось: он был настолько запутанным, что понять, как с ним работать, не получалось. Нельзя было и отложить презентацию: ее уже дважды переносили, так что пришлось показывать программу как есть, вместе с ошибкой. Во время демонстрации генеральный директор был не в духе: эту функциональность предполагалось развернуть еще в прошлом квартале. А тут еще эта ошибка в расчете комиссий так и бросалась в глаза: расхождения в значениях были огромными. Ребята попали под раздачу. Генеральный потребовал от них не просто идти исправлять, а «опрометью и бегом» — прямо так и сказал.

В процессе отладки они все-таки нашли фрагмент кода, в котором комиссии вычислялись с ошибкой, но почему — оставалось загадкой. Эдриенн и ее коллегам пришлось работать сверхурочно и в понедельник, и во вторник, и в среду, и, наконец, им удалось выявить проблему. Все дело было в «умном» коде с однобуквенными именами переменных. И в довершение всех бед выяснилось, что новая программа получала суммы из сгенерированного PDF-файла, а не из более надежного источника, что было серьезным отступлением от соглашений, принятых в кодовой базе, к тому же нерабочим. Стало понятно, что новая функциональность занесла в код несколько значительных, трудно исправляемых багов.

Чтобы наладить расчеты, команда работала в поте лица. Им было непросто: отсутствовали тесты (как же так, Майк?), а код был таким запутанным, что все единодушно решили, что даже Майк не сможет в нем разобраться, когда вернется из отпуска. Эдриенн, Эрика и Джастин сказали друг другу: «Такое больше не должно повториться».

Читая эту историю, вы, возможно, подумали: «Почему же коллеги Майка не просмотрели код перед загрузкой в демосреду? Почему Майк и члены команды не объясняли друг другу сделанные изменения и не применяли автоматические тесты для поиска багов?» Отличные вопросы: они подводят нас к главной теме этой книги — код-ревью.

Так что же такое *код-ревью* (code review)? В целом это процесс, в ходе которого разработчики проверяют написанный коллегами код на соответствие определенным принятым в их команде стандартам. Происходить это может по-разному: в узком кругу за монитором, на общих формальных встречах с участием ревьюеров-специалистов или при помощи *пул-реквестов* (pull request). И в этой книге мы будем говорить в основном о них. Пул-реквест[1] — это предложение по изменению кода, которое нужно проверить и обсудить до загрузки в кодовую базу. Пул-реквесты часто используются в инструментах, подобных GitHub, GitLab, Bitbucket, Azure Repos, AWS CodeCommit, Google Cloud Source Repositories и других системах для совместной работы над кодом на базе частных Git-репозиториев.

ПРИМЕЧАНИЕ Узнать о том, что такое Git, можно из этой статьи: https://mng.bz/w5Z5.

Код-ревью — это как раз то, чего не хватило команде в истории про Майка. Они могли бы использовать простейший вариант этого процесса: Майк подготовил бы пул-реквест с новым кодом, после чего двое, а в идеале — все трое его коллег просмотрели бы изменения и убедились в отсутствии багов. Поскольку ясно, что баги были, Майк получил бы обратную связь: для такой важной функции код слишком сложный, не соответствует принятым соглашениям и не покрыт тестами. И код Майка никогда бы не попал в демосреду.

Нельзя не отметить, что код-ревью дополняют и взаимно переплетаются со стратегией *непрерывной интеграции* (continuous integration, CI) и *непрерывной доставки* (continuous delivery, CD). CI — это автоматизация сборки, тестирования и интеграции изменений в коде в общий репозиторий, CD — автоматизация доставки этих изменений в среду для утверждения. Вместе они образуют *конвейер CI/CD* — автоматизированный процесс, применяемый разработчиками для сокращения части ручных действий по развертыванию. Полностью автоматический конвейер — заманчивая цель, к которой нам стоит стремиться, однако он не способен защитить кодовую базу от багов и поддерживать ее работоспособность. Там, где CI может лишь провести статический анализ, выполнить юнит-тесты и другие доступные компьютеру проверки кода, код-ревью позволяет взглянуть на него с точки зрения человека — опытного разработчика, знакомого с массой нюансов, прекрасно ориентирующегося в контексте и предметной области. Нам нужно и то и другое. В конце концов, чем больше средств мы сможем применить для обеспечения надежности, тем лучше.

---

[1] Иначе его называют запросом на слияние, запросом на внесение изменений или запросом на проверку. — *Примеч. ред.*

Хороший процесс обзора кода играет критически важную роль в общем подходе организации к достижению безопасности цепочки поставок и среды сборки. Вовлеченность каждого члена команды и поддержка код-ревью способствуют общей цели — созданию программного обеспечения, которое приносит пользу людям и прибыль бизнесу.

Немного о непрерывном развертывании

Когда речь заходит о конвейерах CI/CD, сокращение «CD» обычно означает *непрерывную доставку* (continuous delivery), а не *развертывание* (deployment). При непрерывной доставке конвейер в какой-то момент останавливается и ожидает решения человека, который должен одобрить перенос кода в продакшен, тогда как непрерывное развертывание выполняется автоматически, без участия людей. Можно предложить такое сравнение: доставка ждет одобрения (как курьер, который просит расписаться за посылку), развертывание передает пользователям готовый к работе код (в полном соответствии со значением слова «развертывать» — приводить что-либо в положение готовности).

Пожалуй, кто-то из вас может сказать:

— Я знаю, насколько важны код-ревью. И в нашей группе уже налажен этот процесс.

— Прекрасно! — отвечу я. — Отличное начало. Тогда продолжим: я предложу вам несколько утверждений, а вы решите, согласны ли с ними вы, а главное — ваши коллеги. Итак:

- процесс код-ревью приносит нам много пользы;
- просмотр кода не отнимает у нас много времени;
- проверками занимаются несколько человек;
- после горячего обсуждения мы не таим обид друг на друга;
- мы умеем критиковать и вносить предложения конструктивно;
- мы знаем, как эффективно использовать обратную связь;
- процесс код-ревью соответствует целям, которые мы перед ним ставим;
- мы все одинаково понимаем эти цели;
- мы пользуемся преимуществами средств автоматизации;
- мы знаем, что ожидается от всех участников процесса обзора кода;
- мы понимаем важность проверок как части процесса разработки;
- мы можем рассказать другим о пользе код-ревью и его значении для бизнеса;
- мы можем с уверенностью сказать, что *любим* код-ревью.

Я разговаривала с разработчиками из разных стран, да и сама работала в разных командах, а потому уверена, что вы не согласились хотя бы с одним из этих утверждений. И я могу сказать, что это обычное дело: разработчики *боятся* код-ревью. Может быть, потому что считают процесс непоследовательным, нестабильным, неэффективным, несправедливым, недостаточно автоматизированным или иным образом несовершенным или же не питают достаточной симпатии к коллегам. Возможно, тут дело в каких-то других, чисто человеческих моментах, но так или иначе проверки — это настоящее проклятие для очень многих разработчиков.

Что ж, такова реальность. Но с помощью этой книги я бы хотела все изменить. Пусть все: и те, кто впервые слышит о код-ревью, и те, кто давно устал от них, посмотрят на них с правильной точки зрения, увидят их такими, какими они должны быть: эффективными, сплачивающими команду и не требующими значительных усилий.

Однако важно понимать: эта книга — не универсальная инструкция, которая подойдет всем. У каждой команды свои потребности, пожелания и возможности, никто не в силах описать все варианты. Однако вы держите в руках руководство к действию, которое поможет построить или улучшить, а затем постоянно совершенствовать и развивать пригодный к работе процесс код-ревью, который будет функционировать в рамках сложившихся ограничений, позволит решать поставленные задачи, дополнит имеющийся конвейер CI/CD и станет предметом гордости вашей команды. В конечном счете ваш процесс обзора кода улучшится, пусть и не сразу, а постепенно.

Я постараюсь обобщить свои советы и сделать их максимально универсальными. И если сегодня вы последуете хотя бы нескольким из них или поделитесь ими с коллегами, я буду считать это успехом для всех нас. Приступим же к делу.

1.1. Кому адресована книга

Знакомство с книгой будет полезно всем тем, кто читает, пишет и проверяет код. Полезные советы, тактики автоматизации, стратегии и способы командной работы, которые вы найдете на этих страницах, — все это поможет разработчикам, техническим менеджерам и директорам построить с нуля или улучшить имеющийся, но недостаточно эффективный процесс код-ревью.

- *Если вы — начинающий разработчик*, я постараюсь вам объяснить, что обзоры кода важны и жизненно необходимы всем, кто стремится писать понятные, простые в сопровождении, полезные приложения; что рецензирование кода пригодится в любой команде (хоть это и не всегда очевидно ее участникам) и что нельзя просто поручить всю работу ИИ.
- *Если вы понимаете важность код-ревью, но работаете в команде, где его не используют*, я помогу вам не только убедить коллег внедрить проверки кода,

но и построить с нуля справедливый процесс, эффективно решающий как ваши задачи, так и командные.

- *Если вы видите, что применяемый вами процесс обзора кода требует улучшений,* я предложу вам идеи, тактики и стратегии, реализуя и пробуя которые можно решить проблемы, наиболее часто возникающие при просмотре коде. А если под улучшением вы понимаете полную переделку (😉), я посоветую вам, как сформулировать правила для всей команды.
- *Если вы — разработчик, которому во время код-ревью трудно прийти к компромиссу с коллегами*, я предложу вам пошаговую инструкцию; она поможет найти устраивающее всех решение, а также стратегии разрешения типовых разногласий.
- *Если вам трудно писать отзывы на чужой код*, я покажу вам универсальные подходы, которые помогут организовать и упростить общение между автором и ревьюером так, чтобы обе стороны не теряли объективности и работали эффективно.
- *Если вы — сеньор- или лид-разработчик, уставший быть единственным ревьюером в команде*, я помогу вам устранить это узкое место и справедливо распределить ответственность между коллегами так, чтобы процесс обзора кода не только работал по прямому назначению, но и вносил вклад в развитие навыков членов вашей команды.
- *Если вы — техлид или технический менеджер, обеспокоенный поведением членов команды во время код-ревью*, я расскажу вам, как научить коллег писать эффективные, но тактичные отзывы, используя шаблоны комментариев. Кроме того, я познакомлю вас с регламентом проверки кода — документом, который позволит участникам процесса лучше понять, на что следует обращать внимание, делая обзор кода.
- *Если вы — техлид или технический менеджер, расстроенный тем, что на просмотр кода уходит много времени*, я назову вам главные причины задержек и научу находить возможности для ускорения.
- *Если вы — техлид или технический менеджер, желающий привить культуру просмотра кода внутри компании или укрепить ее*, я надеюсь, что эта книга станет для вас полезным руководством по построению целостного процесса проверки кода, даже если вы сами не будете в этом участвовать. Я помогу вам оценить преимущества код-ревью, предвидеть (и решить) распространенные проблемы, предотвратить появление узких мест, осознать, как лучше всего подойти к данной задаче с учетом потребностей и возможностей вашей команды.

Кем бы вы ни были, я покажу вам код-ревью «с человеческим лицом», чтобы они перестали казаться чем-то, о чем стараются поскорее забыть, что ненавидят или используют, чтобы потешить свое самолюбие, чтобы они стали частью рабочего процесса — приятной, желанной и поддерживаемой всей командой. Рецензирование кода — неотъемлемая часть разработки программного обеспечения. Любой

процесс можно улучшить, а проверку кода — тем более, особенно если применять какие-то из бессчетного числа специальных инструментов, имеющихся в нашем распоряжении. И эта книга поможет вам в этом.

Но все-таки нужно сказать: эта книга будет полезна *не всем*. Прежде всего она едва ли поможет тем из вас, кто работает в одиночку (театр одного актера), или же тем, чей стартап еще не дорос до командного уровня. Решение разногласий между людьми — вот главная суть советов и тактик, которыми я поделюсь, а значит, в подобных случаях они неприменимы. Впрочем, они обязательно пригодятся вам, если когда-нибудь вы окажетесь в более многочисленном коллективе.

Кроме того, в этой книге я говорю о том, *как* построить процесс, чтобы проверка кода была эффективной. И это едва ли поможет тем, кто хочет узнать, *что* проверять, какие запахи кода искать. В этом вопросе все сильно зависит от языка программирования, архитектуры конкретного приложения, а также от внутренних стандартов компании. Поэтому если вы ищете справочник по соглашениям и лучшим практикам программирования или учебник, который научит вас писать хороший код, то эта книга вряд ли поможет вам достичь мастерства и совершенства в программировании, каких в военном искусстве достигли непобедимые джедаи в «Звездных войнах».

Если вам нужно именно это, рекомендую книги «Code Complete: A Practical Handbook of Software Construction» Стива Макконнелла (Steve McConnell)[1] [1] и «Five Lines of Code. How and When to Refactor» Кристиана Клаузена[2] (Christian Clausen) [2]. Это отличные справочники, в которых рассказано, на что обращать внимание, рецензируя код, и как можно его улучшить.

1.2. Как устроена книга

В части 1 (главы 1–3) мы начнем с основ. Сейчас вы читаете главу 1 — введение, рассказывающее о преимуществах, которые дают код-ревью, и их значении для разработки программного обеспечения. Здесь же описана структура книги и дан портрет потенциального читателя. В главе 2 мы разберем обзор кода на компоненты и выберем самые важные из них: обсудим разновидности систем и структуру типовых процессов проверки кода, поговорим об основных участниках и зонах их ответственности. В главе 3 речь пойдет о том, как шаг за шагом построить ваш первый процесс просмотра кода, что нужно обсудить с коллегами, какие важные решения принять.

Так, заложив основы, мы перейдем к части 2 (главы 4–6), в которой будем говорить о специфических навыках, необходимых для качественных обзоров кода.

---

[1] Макконнелл С. «Совершенный код. Практическое руководство по разработке программного обеспечения».

[2] Клаузен К. «Пять строк кода». СПб.: издательство «Питер».

В главе 4 мы познакомимся с важным документом — регламентом проверки кода — и рассмотрим все, что с ним связано: от целей и задач до пользы в деле установления и соблюдения правил и политик код-ревью. Глава 5 посвящена системам автоматизации, которые применяются как при написании, так и при рецензировании кода. Из главы 6 мы поймем, какое значение имеют слова и как практически в любых случаях писать тактичные, но эффективные комментарии к пул-реквестам.

Часть 3 (главы 7–10) познакомит вас с трудностями, которые поджидают команду, уже наладившую процесс код-ревью. Из главы 7 мы узнаем о нескольких болевых точках, рассмотрим реальные примеры, когда что-то идет не так. Глава 8 посвящена проблеме затянувшихся проверок кода: поговорим, почему это происходит и как это преодолеть. В главе 9 речь пойдет о лазейках в процессе: я научу вас находить и быстро закрывать их (техлиды и технические менеджеры, эта глава — для вас). И наконец, в главе 10 я расскажу об аварийном плане — полезном инструменте, который поможет вам справляться с ситуациями, в которых нужно отступить от обычного порядка. Иными словами, в части 3 вас ждут действенные советы, которые, возможно, пригодятся вам уже сегодня.

Из части 4 (главы 11–13), прежде чем я оставлю вас наедине с собой поразмышлять о будущем, вы сможете узнать о том, как сочетать код-ревью с другими процессами и практиками разработки. В главе 11 мы рассмотрим парное, а в главе 12 — моб-программирование. Обе главы отвечают на два вопроса: «Нужны ли нам обзоры кода при таких подходах к работе?» и «Как делать хорошо и то и другое?» И наконец, в главе 13 мы затронем тему искусственного интеллекта, его применимости к код-ревью: поговорим об имеющихся (на момент написания книги) возможностях и о том, о чем следует помнить (непременно), используя ИИ для рецензирования.

Я думаю, лучше всего прочитать всю книгу от корки до корки. Впрочем, можно действовать и иначе.

- *Если вы только что загорелись идеей о код-ревью и вам непонятно, с чего начать*, прочтите главу 2, после чего внимательно изучите главы 3 и 4. Когда вы хорошо освоите этот материал, а обзоры кода станут частью вашей работы, прочтите остальные разделы части 3. Они помогут вам улучшить процесс и адаптировать его к конкретным условиям.
- *Если вы испытываете трудности с плохо работающим процессом проверок*, я дам вам тот же совет. Начните с чистого листа (поверьте, неэффективный процесс лучше сначала разрушить): прочтите главу 2, а затем главы 3 и 4. Поскольку у вас имеется пусть *негативный*, но все-таки опыт, скорее всего, вы сможете перейти к оставшимся частям книги быстрее, чем те, кто раньше не практиковал рецензирование: все же вы будете знать, что нужно улучшить или (на этот раз) сделать правильно.

- *Если ваши коллеги осознают, что код-ревью — это важно, однако никак не могут договориться друг с другом*, начните с главы 4 и прочитайте все до конца.
- *Если вы в целом довольны процессом обзора кода, но знаете, что может быть еще лучше*, переходите сразу к главам 5, 6 и последующим, начинайте эксперименты с автоматикой и политиками, а также другими перспективными способами улучшения. При необходимости возвращайтесь к главам 2–4.
- *Если вам и коллегам необходима помощь с конструктивными комментариями (как их писать и как принимать)*, начните чтение с главы 6, к другим же главам обращайтесь по мере необходимости.
- *Общее правило*: главы 2–4 одинаково полезны большинству команд, независимо от того, как далеко вы продвинулись в код-ревью.

И последнее: я полагаю, что эта книга не для разового чтения. Наладив процесс просмотра кода, вы можете возвращаться к ней по мере необходимости. Рекомендуйте отдельные главы новым коллегам или же сделайте книгу обязательной к чтению всеми, кто занят рецензированием. Пробуйте разные тактики, оттачивайте подходы. А эта книга станет вам лучшим помощником.

1.3. Вам нужны код-ревью

Код-ревью должны быть не исключением, а нормой. Они улучшают приложения, делают кодовую базу чище и понятнее, создают бесценные артефакты, при помощи которых легко разобраться в том, *как*, а главное, *зачем* она изменялась. Поверьте: вам обязательно нужно проверять свой код.

1.3.1. Улучшение приложений

Код-ревью (вместе с грамотной стратегией CI/CD) — залог правильной работы и безопасности кодовой базы. При надлежащем и последовательном подходе количество багов, попадающих в продакшен, сводится к минимуму, что было доказано уже много-много раз. В проекте Orbit (около полумиллиона строк кода) компания IBM применила 11(!) уровней проверки кода, и это дало удивительный результат: всего 1 % всех ошибок был обнаружен иными средствами, без код-ревью [1]. В одном из исследований компания AT&T установила, что внедрение практики просмотра кода в отделе со штатом в 200 человек уменьшило число дефектов на 90 % [1]. Крупное исследование по современным код-ревью и безопасности в проектах с открытым исходным кодом, которое провел Калифорнийский университет, показало, что просмотр кода снижает количество багов и дыр в безопасности в продакшене [3]. Но лучше всего, как мне кажется, сказано в статье «The Impact of Code Review Coverage and Code Review Participation on Software Quality» («Влияние покрытия и вовлеченности в код-ревью на качество программных продуктов»): «Большое количество внесенных в процессе разработки изменений, которые (1) вообще не проходили через

код-ревью (низкое покрытие) либо (2) проходили, но недостаточно тщательно (низкая вовлеченность), способствует попаданию в программный продукт потенциально дефектного кода» [4].

Обзоры кода также благоприятно влияют на его чистоту и читабельность. Разработчики проверяют работу друг друга, следовательно, каждый из них, либо из собственной склонности к более понятному коду, либо из уважения к тем, кто будет его читать, стремятся писать аккуратнее, выделяя вносимые изменения. Чистый, легко читаемый и понятный другим код проще сопровождать, а это значит, что в будущем в кодовой базе появится меньше багов. Код-ревью — это важный инструмент контроля, который выполняется только людьми, то есть обеспечивает высокое качество проверки и делает приложения лучше.

Качество кода будет постоянно расти (и оставаться высоким), таким образом он станет понятнее и проще в сопровождении, снизятся затраты времени, денег и человеческих ресурсов, необходимые для отладки и устранения ошибок.

Наличие кодовой базы, с которой можно уверенно работать, упрощает (а иногда и делает возможным) соблюдение требований как нормативных стандартов, так и внутреннего аудита. Со временем во многом благодаря код-ревью сопровождаемость кодовой базы будет только улучшаться. Все это способствует повышению качества кода, а значит, и приложений.

1.3.2. Углубление взаимопонимания

Нельзя недооценивать и то, что благодаря проверкам кода члены команды начинают лучше понимать и кодовую базу, и друг друга. При должном отношении обзоры кода работают как механизм передачи знаний (целенаправленно от одного человека к другому) и обмена ими (в свободной и удобной форме). А это облегчает онбординг новых разработчиков и их интеграцию в команду.

Все участники рецензирования естественным образом вовлекаются в процессы обмена и передачи знаний и знакомятся с гораздо большей частью кодовой базы. При этом распространение знаний снижает зависимость команды от отдельных людей, а следовательно, и нагрузку на них. Вам больше не придется винить себя за то, что взяли отгул.

Наконец, эффективные код-ревью в сочетании с автоматизацией и четкой политикой команды (например, правильные сообщения о коммитах, описания пул-реквестов и обязательные объяснения причин, по которым происходят изменения) могут стать хроникой развития кодовой базы. Представьте себе, как полезно знать, когда, где и зачем что-то было изменено и привело к проблемам или, наоборот, принесло пользу.

Внедрение код-ревью дает множество важных преимуществ, и по идее все должны это понимать. Вот только, увы, находятся те, кто не разделяет это мнение.

1.4. Убеждаем коллег

Давайте уясним: большинство разработчиков не против код-ревью как таковых. Их раздражают некоторые сопровождающие процесс явления. Поэтому нужно готовиться к тому, что вам придется убеждать и уговаривать своих коллег, особенно если все начинается с нуля! Вот почему так важно прежде всего найти с ними общий язык (прочитав вместе эту книгу 😉).

Как поется в песне Халида, «Can't we just talk?» («Может, просто поговорим?»). Агитация коллег начинается с открытого разговора с ними, бесед, в которых должны участвовать все, от пишущих код разработчиков до технических менеджеров, формирующих политики, и даже технических директоров, которые будут эти политики внедрять.

Во время такой беседы (а может, и нескольких) необходимо обсудить и согласовать множество вопросов. Прежде всего:

- Каковы цели код-ревью?
- Кому нужно знать об изменениях?
- Кто будет решать, кому проверять пул-реквест?
- Сколько времени ревьюер может взять на проверку?
- Сколько одобрений должен получить пул-реквест: два или, может быть, три?
- Что делать, если необходимо развернуть аварийный хотфикс?

Сначала такие беседы можно проводить еженедельно, к примеру, на технологических совещаниях, которые как раз и бывают раз в неделю и где обсуждаются планы, проекты и цели. По мере формирования и уточнения правил работы нужно фиксировать их в отдельном документе, который в дальнейшем использовать как руководство к действию. К примеру, можно составить регламент проверки кода, речь о котором пойдет в главе 4.

Узнайте мнение каждого из коллег, активно поощряйте их к участию в обсуждениях, выстраивайте процесс код-ревью, исходя из общих целей. И рано или поздно вы сумеете убедить команду внедрить код-ревью.

1.5. Улучшаем код-ревью

Как только в вашей команде будет внедрена практика код-ревью, ваши коллеги, скорее всего, захотят внести улучшения в процесс (а может быть, вы и стали читать эту книгу именно с такой целью). Для этого нужно будет учесть два очень важных момента.

Во-первых, инженерная дисциплина должна быть нормой для команды. В свете код-ревью это означает создание артефакта (исходного кода), который

отличается высокой долговечностью и ценностью. Если разработчики в вашей команде легко понимают написанный ими код и смогут понять его даже несколько лет спустя, можно считать, что он долговечен. Ценность же кода тем выше, чем меньше времени уходит на устранение ошибок, чем проще расширить функциональность. В команде, где прочно укоренились такие принципы, и качество кода, и эффективность процессов код-ревью, скорее всего, находятся на высоте.

Во-вторых, не следует забывать о людях, пишущих код. Прекрасно работающий процесс обзоров кода — от применяемых методов до обсуждений — можно наладить только в сплоченной команде, где поощряется обмен мнениями. А какую команду можно считать сплоченной? Такую, где каждый из коллег хочет и может делиться с другими своим мнением, участвовать в общих обсуждениях, не боится говорить о проблемах и может быть полностью уверен в том, что другие члены команды делают свою часть работы с полной отдачей, а лично к нему относятся доброжелательно и с уважением. В сплоченной команде нет места собственному эго, предвзятости, протекционизму, кумовству, себялюбию и гордыне.

Чтобы построить успешный процесс код-ревью, команда должна понимать, чего хочет, а чего нет, какие системы лучше всего подойдут для нее, что можно сделать при существующих ограничениях (на уровне команды и организации), каковы общие цели и стандарты. Кроме того, необходимо иметь коллективное чувство ответственности за кодовую базу, а это происходит лишь тогда, когда команда работает слаженно, как единый механизм. Только при этих условиях можно построить процесс рецензирования, способный стабильно приносить полезные результаты.

Ну а теперь я хочу поделиться с вами всем тем, что я сама испытала, узнала, придумала, вывела в теории и проверила опытным путем, в чем я убеждалась в течение многих лет: как сделать код-ревью настолько хорошими, насколько это вообще возможно. И моя книга — набор идей, тактик, шаблонов, паттернов и стратегий, при помощи которых вы сможете довести свой процесс до идеала — кульминация всей проделанной работы. Готовы?

Итоги

- Код-ревью — это процесс, в ходе которого разработчики проверяют написанный коллегами код на соответствие определенным принятым в команде стандартам. Проверка может происходить в узком кругу за монитором, на общих формальных встречах или при помощи пул-реквестов (предложений по изменению кода, которые можно проверить и обсудить до слияния в общую ветку).
- Код-ревью должны быть не исключением, а нормой. Почему? Потому что благодаря им приложения становятся лучше (за счет более чистого, читабельного и сопровождаемого кода), а люди из работающей над ними команды

глубже понимают и кодовую базу, и друг друга (за счет обмена знаниями, а также ведения записей об изменениях).

- Слаженно работающие процессы проверки кода создаются в сплоченных командах, где каждый знает, что его уважают и ценят, и где царит дух равноправного сотрудничества.

Ссылки

[1] McConnell, S. (2016). *Code Complete: A Practical Handbook of Software Construction*.[1] Microsoft Press.

[2] Clausen, C. (2021). *Five Lines of Code: How and When to Refactor*.[2] Manning Publications.

[3] Thompson, C., & Wagner, D. (2017). *A large-scale study of modern code review and security in open source projects*. In Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering (pp. 83–92). Association for Computing Machinery.

[4] McIntosh, S., Kamei, Y., Adams, B., & Hassan, A. E. (2014). *The impact of code review coverage and code review participation on software quality*. In MSR 2014: Proceedings of the 11th Working Conference on Mining Software Repositories (pp. 192–201). Association for Computing Machinery.

[1] Макконнелл С. «Совершенный код. Практическое руководство по разработке программного обеспечения».

[2] Клаусен К. «Пять строк кода». СПб.: издательство «Питер».