ГЛАВА 19

Critique: инструмент обзора кода в Google

Авторы: Кейтлин Садовски, Ильхам Курния и Бен Рольфс
Редактор: Лиза Кэри

Как мы уже писали в главе 9, обзор кода является жизненно важной частью разработки ПО, особенно в больших масштабах. Основная цель обзора кода — улучшить удобочитаемость и простоту поддержки кодовой базы, и достижение этой цели обеспечивается процессом обзора и инструментами, поддерживающими этот процесс.

В этой главе мы посмотрим, как в Google проводится обзор кода с использованием собственного инструмента *Critique*. Он явно поддерживает основные цели рецензирования, давая рецензентам и авторам возможность исследовать и комментировать изменения. Также Critique дает возможность фильтровать код, поступающий в кодовую базу (см. раздел, посвященный «оценке» изменений). Информация из Critique используется при поиске причин технических решений по диалогам в обзорах кода (например, когда отсутствуют встроенные комментарии). Critique — не единственный, но самый популярный инструмент обзора кода в Google.

Принципы оснащения обзора кода инструментами

Выше упоминалось, что Critique предоставляет необходимые функции поддержки целей обзора кода (мы рассмотрим их далее в этой главе), но почему этот инструмент пользуется таким успехом? Critique сформирован культурой разработки в Google, в которой обзор кода является неотъемлемой частью рабочего процесса. Это культурное влияние выражается в наборе руководящих принципов, которые Critique подчеркивает:

Простота
: Пользовательский интерфейс Critique создавался с целью упростить обзоры кода. Он не имеет лишних элементов управления, работает очень гладко, быстро загружается, имеет простую навигацию, поддерживает горячие клавиши, а также четкие визуальные индикаторы, сообщающие, было ли проверено то или иное изменение.

Доверие
: Обзоры кода должны не замедлять работу инженеров, а расширять их возможности. Они основываются на максимальном доверии к коллегам, например к авторам изменений. Расширению доверия способствует также общедоступность (для просмотра и проверки) изменений в Google.

Универсальная коммуникация
: Сложности с коммуникацией редко решаются с помощью инструментов. Critique отдает приоритет универсальным способам комментирования изменений в коде, а не сложным протоколам. Вместо того чтобы усложнять модель данных и процесс, Critique поощряет пользователей пояснять в своих комментариях, чего они хотят, и предлагает правки к комментариям. Но даже с использованием лучшего инструмента обзора кода коммуникация может не получаться, потому что пользователи — это люди.

Интеграция в рабочий процесс
: Critique имеет ряд точек интеграции с другими инструментами разработки ПО. Разработчики с легкостью могут переходить от просмотра проверяемого кода в Code Search к правке в веб-инструменте редактирования кода или к просмотру результатов тестирования изменений.

Из всех этих руководящих принципов наибольшее влияние на инструмент оказала простота. Изначально мы думали о добавлении множества интересных возможностей, но потом решили не усложнять модель, чтобы обеспечить поддержку максимально широкого круга пользователей.

Простота также имеет интересное противоречие с интеграцией Critique в рабочий процесс. Мы рассматривали возможность создания единого инструмента Code Central для правки, обзора и поиска кода, но потом отказались от этой идеи. Critique имеет множество точек соприкосновения с другими инструментами, но мы сознательно решили нацелить его на обзор кода. Многие возможности, доступные в Critique, реализованы в разных подсистемах.

Процесс обзора кода

Обзоры кода могут выполняться на разных этапах разработки ПО (рис. 19.1). Обычно они проводятся до передачи изменений в кодовую базу, поэтому их называют *обзорами перед передачей в репозиторий*. Краткое описание процесса проверки кода уже приводилось в главе 9, но мы повторим его здесь, дополнив некоторыми ключевыми аспектами Critique. В следующих разделах мы подробно рассмотрим каждый этап.

Вот типичные этапы обзора кода:

1. **Добавление изменений.** Пользователь вносит изменение в код в своем рабочем пространстве. Затем *автор* выгружает *снимок* (отражающий изменения в определенный момент) в Critique, который запускает инструменты автоматического анализа кода (глава 20).

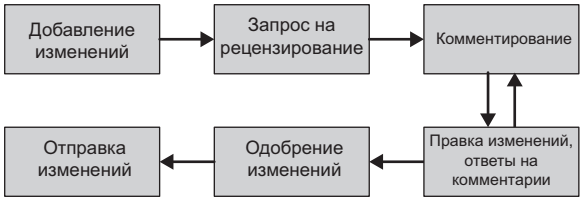

Рис. 19.1. Процесс обзора кода

2. **Запрос на рецензирование.** Закончив вносить изменения и удовлетворившись результатами автоматического анализа в Critique, автор отправляет изменение одному или нескольким рецензентам по электронной почте.
3. **Комментирование.** *Рецензенты* открывают изменение в Critique и добавляют свои комментарии. По умолчанию комментарии отмечаются как *нерешенные*, то есть требующие реакции автора. Также рецензенты могут добавлять *решенные* комментарии, которые не требуют реакции автора и имеют исключительно информационный характер. Результаты автоматического анализа кода, если они есть, также доступны рецензентам. После того как рецензент подготовит набор комментариев, он *публикует* их для автора, выражая свою оценку изменения после его обзора. Прокомментировать изменение может любой желающий в «попутном обзоре».
4. **Правка изменений, ответы на комментарии.** Основываясь на комментариях, автор вносит новые правки в изменение, выгружает новые снимки и передает обновленный вариант изменения рецензентам. Автор реагирует (по меньшей мере) на все нерешенные комментарии, либо изменяя код, либо просто отвечая на комментарий и отмечая комментарий как *решенный*. Автор и рецензенты могут посмотреть различия между любыми парами снимков, чтобы увидеть, что изменилось. Шаги 3 и 4 могут повторяться несколько раз.
5. **Одобрение изменений.** Удовлетворившись последним состоянием изменения, рецензенты одобряют его и отмечают как «мне нравится» (LGTM). При желании они могут добавить комментарии, требующие решения. После признания изменения пригодным для отправки в репозиторий оно будет отмечено зеленым цветом в пользовательском интерфейсе.
6. **Отправка изменений.** Если изменение одобрено (как обсуждается ниже), автор может запустить процесс его фиксации. Если инструменты автоматического анализа и другие средства оценки перед отправкой не обнаружили никаких проблем, изменение фиксируется в кодовой базе.

Даже после запуска процесса система позволяет отклониться от типичной процедуры обзора. Например, рецензенты могут отказаться от оценки изменений или явно передать эту роль кому-то другому, а автор может вообще приостановить процесс обзора. В экстренных случаях автор может принудительно зафиксировать свое изменение в репозитории и запросить его обзор после фиксации.

Уведомления

По мере продвижения изменения через этапы, описанные выше, Critique публикует уведомления о событиях, которые могут использоваться другими вспомогательными инструментами. Модель уведомлений позволяет инструменту Critique оставаться основным средством обзора кода, интегрированным в рабочий процесс разработчика, и не превращаться в многоцелевой инструмент. Уведомления позволяют разделить задачи так, чтобы Critique мог просто посылать события, а другие системы обслуживали их.

Например, пользователи могут установить расширение для Chrome, чтобы подписаться на эти уведомления. Когда появляется изменение, требующее внимания пользователя, например если настала очередь пользователя оценить изменение или предварительная проверка перед отправкой потерпела неудачу, расширение выводит на экран уведомление с кнопкой для перехода непосредственно к изменению или для отключения уведомления. Мы заметили, что одним разработчикам нравятся немедленные уведомления об обновлении изменений, а другие не пользуются этим расширением, потому что оно мешает работе.

Critique может рассылать уведомления по электронной почте. Например, таким способом рассылаются уведомления о наиболее важных событиях. Некоторые инструменты автоматического анализа тоже имеют настройки для пересылки результатов по электронной почте дополнительно к их отображению в пользовательском интерфейсе Critique. Кроме того, Critique может обрабатывать ответы по электронной почте и переводить их в комментарии для тех пользователей, которые предпочитают электронную почту. Обратите внимание, что для многих пользователей электронные письма не являются ключевой функцией обзора кода и для управления отзывами они используют панель инструментов Critique (подробнее ниже).

Этап 1: добавление изменений

Инструмент обзора кода должен обеспечивать поддержку на всех этапах процесса рецензирования и не задерживать фиксацию изменений. На этапе предварительного обзора авторам проще отшлифовать свое изменение перед отправкой на проверку, что помогает сократить время рецензирования. Critique позволяет отображать только значимые различия и игнорировать изменение количества пробелов. Также Critique отображает результаты сборки, тестирования и статического анализа, включая проверку стиля (глава 9).

Critique позволяет автору видеть свои изменения так, как их видят рецензент и автоматический анализ, поддерживает возможность корректировки изменения прямо в инструменте обзора и предлагает подходящих рецензентов. Отправляя запрос, автор может добавить предварительные комментарии к изменению, что дает возможность напрямую задать рецензентам любые вопросы. Все это предотвращает недопонимание.

Чтобы предоставить рецензентам дополнительную информацию, автор может связать изменение с конкретной ошибкой. Critique использует службу автодополнения, чтобы подсказать соответствующие ошибки, отдавая приоритет ошибкам, устранение которых поручено автору.

Представление различий

Главная задача обзора кода — упростить понимание изменения. Крупные изменения обычно труднее понять, чем мелкие. Поэтому представление различий в коде, обусловленных изменением, в наиболее простом и понятном виде является основным требованием для хорошего инструмента обзора кода.

В Critique этот принцип распространяется на несколько уровней (рис. 19.2). Компонент представления различий, начиная с алгоритма оптимизации самой длинной общей подпоследовательности, дополнен следующими возможностями:

- подсветка синтаксиса;
- поддержка перекрестных ссылок (с помощью Kythe, глава 17);
- отображение различий внутри строк, показывающее разницу на уровне символов (рис. 19.2);
- настройки, позволяющие игнорировать различия в количестве пробелов;
- фрагменты кода, перемещенные из одного места в другое, отмечаются как перемещенные, а не удаленные в одном месте и добавленные в другом, как это делают многие алгоритмы сравнения.

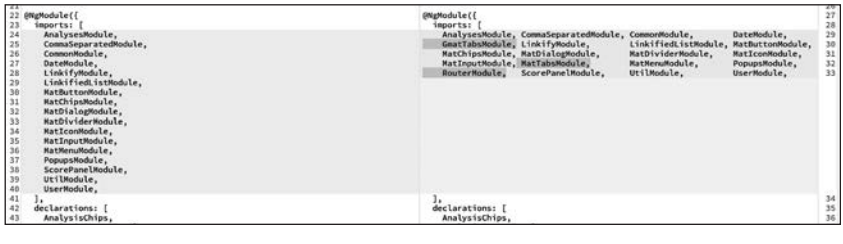

Рис. 19.2. Отображение различий внутри строк показывает разницу на уровне символов

Пользователи также могут просматривать различия в разных режимах, например в режиме наложения или рядом друг с другом. Создавая Critique, мы решили, что важно показать различия рядом друг с другом, чтобы упростить процесс обзора. Такое расположение различий занимает много места, и мы упростили структуру представления различий, отказавшись от границ и отступов и оставив только сами различия и номера строк. Нам пришлось поэкспериментировать с разными шрифтами и кеглями, пока мы не добились представления различий, которое учитывает ограничение размеров строк в 100 символов в Java и типичную ширину экрана в Critique — 1440 пикселов.

Critique также поддерживает дополнительные инструменты, представляющие различия в артефактах, созданных в результате изменения, например снимки пользовательского интерфейса или файлы конфигурации, полученные в результате изменения.

Чтобы упростить процесс навигации по различиям, мы постарались максимально сэкономить пространство и обеспечить быструю загрузку различий, даже таких, как изображения и большие файлы. Мы также определили комбинации клавиш для быстрой навигации по файлам при обзоре измененных разделов.

Когда пользователи переходят на уровень файлов, Critique предоставляет виджет с компактным отображением цепочки версий снимков, с помощью которого пользователи могут выбирать интересующие их версии для сравнения. Этот виджет автоматически сворачивает похожие версии, помогая сосредоточить внимание на наиболее важных из них. Благодаря этому пользователь может видеть, как развивался файл в ходе изменения, например какие части изменения охвачены тестами, а какие части уже проверены или содержат комментарии. Чтобы решить проблему масштаба, Critique заранее выбирает все необходимое, поэтому загрузка различных снимков выполняется очень быстро.

Анализ результатов

Выгрузка снимка приводит к запуску автоматических инструментов анализа кода (глава 20). Результаты и обобщенные итоги анализа отображаются в Critique на странице изменения (рис. 19.3). Более подробные детали отображаются во вкладке Analysis (Анализ) (рис. 19.4).

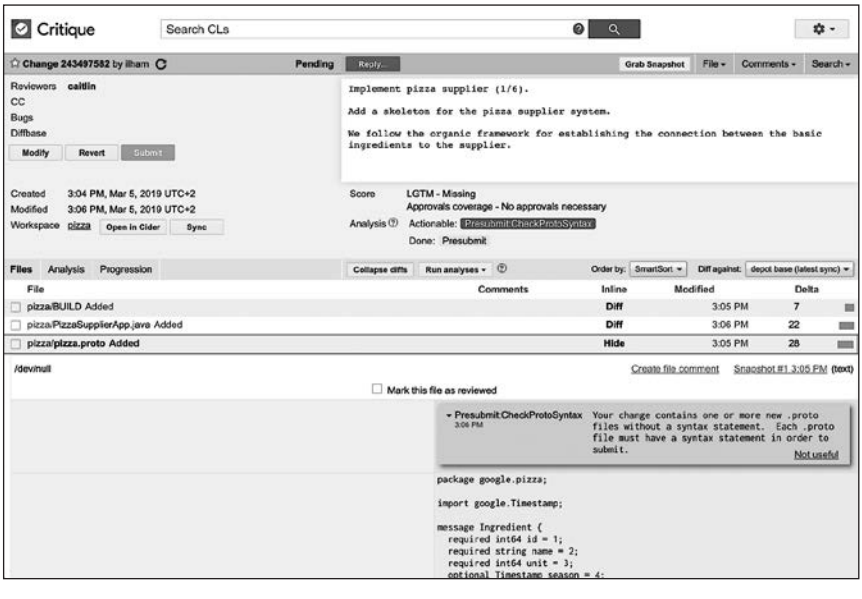

Рис. 19.3. Обобщенные итоги анализа и представление различий

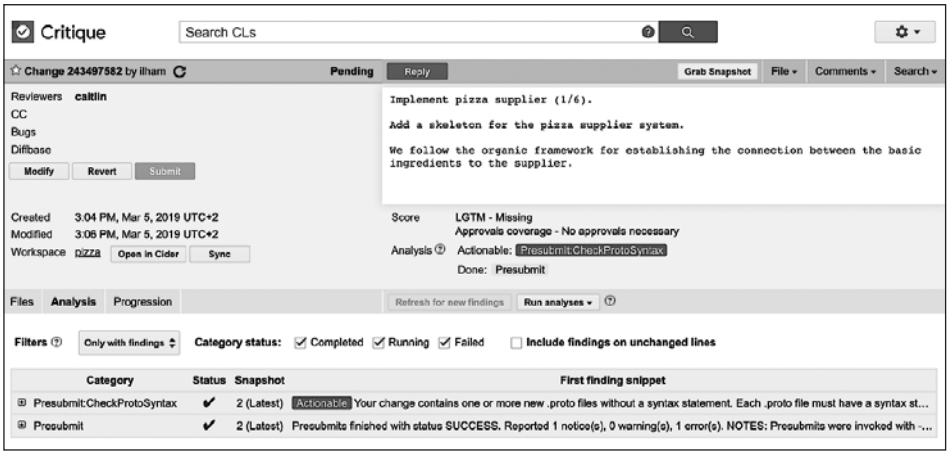

Рис. 19.4. Результаты анализа

Инструменты анализа могут подсвечивать конкретные результаты красным цветом. Еще не завершенные виды анализа выделяются желтыми значками, а остальные — серыми. Для простоты Critique не предлагает других вариантов выделения. Если тот или иной инструмент возвращает какие-то результаты, их можно открыть щелчком мыши на значке. Подобно комментариям, результаты анализа могут отображаться внутри различий, но оформляются иначе, чтобы их было проще отличить. Иногда результаты анализа включают предложения по исправлению замечаний, которые автор может просмотреть и применить прямо в Critique.

Предположим, что инструмент статического анализа обнаруживает нарушение стиля — лишние пробелы в конце строки. На странице изменения появится значок этого инструмента. Щелкнув на нем, автор может быстро перейти к различиям, показывающим код, и парой щелчков мыши исправить нарушение. Большинство замечаний, возвращаемых инструментом статического анализа, также включают предложения по исправлению. Одним щелчком автор может просмотреть предлагаемое исправление (например, предложение удалить лишние пробелы), а вторым — применить его.

Тесная интеграция инструментов

В Google есть инструменты, созданные на основе Piper — монолитного репозитория исходного кода (глава 16), например:

- Cider — онлайн-IDE для редактирования исходного кода, хранящегося в облаке;
- Code Search — инструмент для поиска в кодовой базе;
- Tricorder — инструмент для отображения результатов статического анализа (упоминался выше);

- Rapid — инструмент для выпуска новых версий, который упаковывает и развертывает двоичные файлы, содержащие серии изменений;
- Zapfhahn — инструмент вычисления доли кода, охваченной тестами.

Кроме того, существуют службы, возвращающие информацию об изменениях (например, имена пользователей — авторов изменений или ошибки, для устранения которых создавались изменения). Critique идеально подходит для быстрого доступа к этим системам и организации пользовательского интерфейса к ним, однако мы не используем его для этого, чтобы не жертвовать простотой. Например, на странице изменения в Critique автору достаточно щелкнуть только один раз, чтобы начать редактирование изменения в Cider. Существует также поддержка навигации по перекрестным ссылкам Kythe и просмотра кода в главной ветви с помощью Code Search (глава 17). Critique поддерживает связь с инструментом, показывающим, относится ли отправленное изменение к конкретной версии. В Critique отдается предпочтение ссылкам, а не встроенным операциям, чтобы не отвлекать пользователей от основного процесса обзора кода. Единственным исключением является оценка охвата тестами: информация об охвате строк кода тестами отображается разными цветами фона на полях в представлении различий (не все проекты используют отдельный инструмент для оценки охвата тестирования).

Обратите внимание, что тесная интеграция Critique с рабочим пространством разработчика во многом возможна благодаря хранению рабочих пространств в распределенной файловой системе FUSE, расположенной за пределами компьютера конкретного разработчика. Источник истины находится в облаке и доступен для всех инструментов.

Этап 2: запрос на рецензирование

После того как автор доведет изменение до желаемого состояния, он может передать его для обзора (рис. 19.5). Найти рецензента в небольшой команде может показаться простой задачей, но даже в таком случае желательно равномерно распределить рецензирование между членами команды и учитывать такие ситуации, когда кто-то находится в отпуске. Для этого можно создать отдельный электронный адрес, куда будут направляться все запросы на рецензирование, после чего инструмент *GwsQ* (названный в честь команды Google Web Server, первой предложившей этот прием) на основе своих настроек выберет рецензентов из списка для выполнения обзора.

Учитывая размер кодовой базы в Google и количество инженеров, постоянно изменяющих ее, иногда трудно определить, кто вне вашего проекта лучше подходит для обзора изменений. Поиск рецензентов — сложная задача, которую придется решать, достигнув определенного масштаба кодовой базы. Critique должен поддерживать работу в масштабе и обладает функциональными возможностями для выбора рецензентов, обладающих достаточными полномочиями для одобрения изменения. Функция выбора рецензентов учитывает следующие факторы:

- кому принадлежит изменяемый код (см. следующий раздел);
- кто лучше всего знаком с кодом (то есть кто недавно его изменял);
- кто может заняться обзором (то есть находится на рабочем месте и желательно в том же часовом поясе);
- настройки GwsQ в команде.

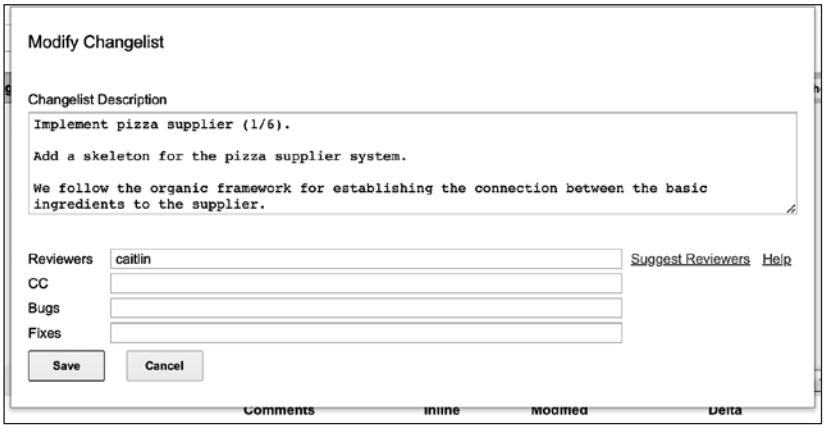

Рис. 19.5. Отправка запроса на рецензирование

Выбор рецензента для обзора изменения инициирует отправку запроса на рецензирование. Этот запрос попадает в обработчик «предварительной проверки», подходящий для изменения: команды могут по-разному настраивать обработчики, связанные с их проектами. Чаще всего используются обработчики:

- автоматически добавляющие изменение в списки рассылки для повышения осведомленности и прозрачности;
- выполняющие автоматизированные наборы тестов для проекта;
- добавляющие постоянные элементы в код (для соблюдения локальных ограничений оформления кода) и в описание изменения (для создания примечаний к версии и добавления другой информации).

Поскольку для тестирования требуются значительные ресурсы, в Google тесты выполняются в ходе предварительной проверки (запускаются при отправке запроса на рецензирование и при фиксации изменений в репозитории), а не для каждого снимка, как, например, анализ с помощью Tricorder. Critique отображает результаты выполнения обработчиков, подобно результатам инструментов анализа, но с важным отличием — неудачный результат блокирует отправку изменения на обзор или его фиксацию. Critique уведомляет автора по электронной почте, если предварительная отправка изменения не удалась.