**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Гжельский государственный университет»** (ГГУ)

Колледж ГГУ

Специальность 09.02.07. Информационные системы и программирование.

**Реферат**

по предмету « Управление проектами»

на тему «Ревьюирование программного кода code review».

ВЫПОЛНИЛ:

Студент группы ИСП-О-17

Белкин И.В.

ПРОВЕРИЛА:

Прокуронова А.Ю.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

п. Электроизолятор

2019 г.

## Что такое code review

Code review - инженерная практика в терминах гибкой методологии разработки. Это анализ (инспекция) кода с целью выявить ошибки, недочеты, расхождения в стиле написания кода, в соответствии написанного кода и поставленной задачи.  
  
К очевидным плюсам этой практики можно отнести:

* Улучшается качество кода
* Находятся «глупые» ошибки (опечатки) в реализации
* Повышается степень совместного владения кодом
* Код приводится к единому стилю написания
* Хорошо подходит для обучения «новичков», быстро набирается навык, происходит выравнивание опыта, обмен знаниями.

Что можно инспектировать

Для ревью подходит любой код. Однако, review **обязательно** должно проводиться для критических мест в приложении (например: механизмы аутентификации, авторизации, передачи и обработки важной информации — обработка денежных транзакций и пр.).  
Также для review подходят и юнит тесты, так как юнит тесты — это тот же самый код, который подвержен ошибкам, его нужно инспектировать также тщательно как и весь остальной код, потому что, неправильный тест может стоить очень дорого.  
  
Как проводить review

Вообще, ревью кода должен проводиться в совокупности с другими гибкими инженерными практиками: парное программирование, TDD, CI. В этом случае достигается максимальная эффективность ревью. Если используется гибкая методология разработки, то этап code review можно внести в Definition of Done фичи.  
  
Из чего состоит review

* Сначала **design review** — анализ будущего дизайна (архитектуры).Данный этап очень важен, так как без него ревью кода будет менее полезным или вообще бесполезным (если программист написал код, но этот код полностью неверен — не решает поставленную задачу, не удовлетворяет требованиям по памяти, времени). Пример: *программисту поставили задачу написать алгоритм сортировки массива. Программист реализовал алгоритм bogo-sort, причем с точки зрения качества кода — не придраться (стиль написания, проверка на ошибки), но этот алгоритм совершенно не подходит по времени работы. Поэтому ревью в данном случае бесполезно (конечно — это утрированный пример, но я думаю, суть ясна), здесь необходимо полностью переписывать алгоритм*.
* Собственно, сам **code review** — анализ написанного кода. На данном этапе автору кода отправляются замечания, пожелания по написанному коду.

Также очень важно определиться, за кем будет последнее слово в принятии финального  решения в случае возникновения спора. Обычно, приоритет отдается тому кто будет реализовывать код (как в scrum при проведении planning poker), либо специальному человеку, который отвечает за этот код (как в google — code owner).  
  
**Как проводить design review**

**Design review** можно проводить за столом, в кругу коллег, у маркерной доски, в корпоративной wiki. На **design review** тот, кто будет писать код, расскажет о выбранной стратегии (примерный алгоритм, требуемые инструменты, библиотеки) решения поставленной задачи. Вся прелесть этого этапа заключается в том, что ошибка проектирования будет стоить 1-2 часа времени (и будет устранена сразу на review).

### Как проводить code review

Можно проводить code review разными способами — дистанционно, когда каждый разработчик сидит за своим рабочим местом, и совместно — сидя перед монитором одного из коллег, либо в специально выделенным для этого месте, например meeting room. В принципе существует много способов (можно даже распечатать исходный код и вносить изменения на бумаге).  
  
**Pre-commit review**

Данный вид review проводится перед внесением изменений в VCS. Этот подход позволяет содержать в репозитории только проверенный код. В microsoft используется этот подход: всем участникам review рассылаются патчи с изменениями. После того как собран и обработан фидбэк, процесс повторяется до тех пор пока все ревьюверы не согласятся с изменениями.  
  
**Post-commit review**

Данный вид review проводится после внесения изменений в VCS. При этом можно коммитить как в основную ветвь, так и во временную ветку (а в основную ветку вливать уже проверенные изменения).  
  
**Тематические review**

Можно также проводить тематические code review — их можно использовать как переходный этап на пути к полноценному code review. Их можно проводить для критического участка кода, либо при поиске ошибок. Самое главное — это определить **цель** данного **review**, при этом цель должна быть обозримой и четкой:  
\*"*Давайте поищем ошибки в этом модуле*" — не подходит в качестве цели, так как она необозрима.

\*"*Анализ алгоритма на соответствие спецификации RFC 1149*" — уже лучше.

Основное отличие тематических review от полноценного code review — это их узкая специализация. Если в code review мы смотрим на стиль кода, соответствие реализации и постановки задачи, поиск опасного кода, то в тематическом review мы смотрим обычно только один аспект (чаще всего — анализ алгоритма на соответствие ТЗ, обработка ошибок).  
Преимущество такого подхода заключается в том, что команда постепенно привыкает к практике review (его можно использовать нерегулярно, по требованию). Получается некий аналог мозгового штурма. Мы использовали такой подход при поиске логических ошибок в нашем ПО: смотрели «старый» код, который был написан за несколько месяцев до review (это можно отнести тоже к отличиям от обычного review — где обычно смотрят свежий код).  
  
**Результаты review**

Самое главное при проведении review — это использование полученного результата. В результате review могут появиться следующие артефакты:

* Описание способа решения задачи (design review)
* UML диаграммы (design review)
* Комментарии к стилю кода (code review)
* Более правильный вариант (быстрый, легкочитаемый) реализации (design review, code review)
* Указание на ошибки в коде (забытое условие в switch, и т.д.) (code review)
* Юнит тесты (design review, code review)

При этом очень важно, чтобы все результаты не пропали, и были внесены в VCS, wiki. Этому могут препятствовать:

* Сроки проекта.
* Лень, забывчивость разработчиков
* Отсутствие удобного механизма внесения изменений review, а также контроль внесения этих изменений.
* Для преодоления этих проблем частично может помочь:
* pre-commit hook в VCS
* Создание ветви в VCS, из которой изменения вливаются в основную ветвь только после review
* Запрет сборки дистрибутива на CI сервере без проведения review. Например, при сборке дистрибутива проверять специальные свойства (svn:properties), либо специальный файл с результатами review. И отказывать в сборке дистрибутива, если не все ревьюверы одобрили (approve) код.
* Использование методологии в разработке (в которой code review является неотъемлемой частью).

**Сложности при проведении review (субъективное мнение)**

Основная сложность, с которой мы столкнулись, когда внедряли review в первый раз: это невозможность контроля изменений по результатам review. Отчасти это связано с тем, что данная практика применялась без других практик — CI (это еще раз доказывает тот факт, что все инженерные практики должны применяться вместе).

**Утилиты для review**

Вообще, утилит для проведения review существует большое количество, как платных, так и бесплатных. Я не стал их приводить, чтобы не навязывать свою точку зрения, в интернете можно найти множество инструментов и подробное описание.