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 «Моделирование и анализ ПО»

на тему «Цели, задачи, этапы и объекты ревьюирования.  
Планирование ревьюирования»

ВЫПОЛНИЛА:

Студентка группы ИСП-О-17

Валгуцкова С.С.

ПРОВЕРИЛА:

Прокуронова А. Ю.

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

п. Электроизолятор

2019 год

**Инспекция кода (Code review)** – систематический и периодический анализ программного кода, направленный на поиск необнаруженных на ранних стадиях разработки программного продукта ошибок, а также, на выявление некачественных архитектурных решений и критических мест в программе.

**Планирование.**

По завершению процесса инициализации ведущий проверяет, что инспектируемые документы размещены в базе данных проекта, а их статус соответствует готовности к формальной инспекции. Если это не так, инспекция откладывается.

Затем ведущий должен изменить статус инспектируемых документов так, чтобы отметить факт начала инспекции и ограничить доступ к инспектируемой документации. Во время инспекции изменение документов невозможно, а соответствующий статус сохраняется до конца инспекции. Этот статус называется Review.

После этого ведущий должен скопировать из базы данных проекта бланк инспекции и занести в него идентификаторы инспектируемых и исходных документов и номера их версий, список участников с указанием их ролей и дату фактического начала процесса инспекции, т.е. того момента, когда инспектируемые документы были переведены в состояние Review

# **Стандарт код-ревью**

Основная цель код-ревью заключается в том, чтобы гарантировать постоянное улучшение кодовой базы Google. Все инструменты и процессы посвящены этой цели.  
Здесь необходим ряд компромиссов.  
Во-первых, разработчики должны быть в состоянии успешно *решать свои задачи*. Если вы никогда не отправляете код, то и кодовая база никогда не улучшится. Кроме того, если рецензент сильно затрудняет *любую* работу, то в будущем разработчики не заинтересованы предлагать улучшения.  
С другой стороны, обязанность рецензента убедиться, что качество CL не снизит общее качество кодовой базы со временем. Это может быть сложно, потому что часто деградация происходит из-за небольшого снижения качества кода со временем, особенно если команда находится под сильным давлением сроков и чувствует, что имеет право на увеличение технического долга.  
Кроме того, рецензент несёт ответственность за рецензируемый код. Он хочет убедиться, что кодовая база остаётся последовательной, поддерживаемой и соответствует всему остальному, что упомянуто в разделе [***«Что проверять в коде»***](https://habr.com/ru/post/467039/#2).  
Таким образом, мы получаем следующее правило в качестве стандарта для код-ревью:  
**Обычно рецензенты должны одобрить CL, как только он достигает состояния, когда определённо улучшает общее качество кода системы, даже если CL не идеален.**  
Это *главный* среди всех принципов код-ревью.  
Конечно, у него есть ограничения. Например, если CL добавляет функцию, которую рецензент не хочет видеть в системе, то рецензент, безусловно, может отказать в коммите, даже если код хорошего качества.  
Ключевым моментом здесь является то, что не бывает «идеального» кода — бывает только код *получше*. Рецензент не должен требовать от автора полировать каждый крошечный фрагментик. Скорее, рецензент должен сбалансировать необходимость дальнейшего прогресса по сравнению с важностью предлагаемых изменений. Вместо того, чтобы стремиться к идеалу, рецензент должен стремиться к *непрерывному улучшению*. Коммит, который в целом улучшает ремонтопригодность, читаемость и понятность системы, нельзя задерживать на дни или недели, потому что он не «идеален».  
Рецензенты *всегда* могут оставлять любые комментарии по улучшению кода, но не очень важные изменения следует помечать, например, префиксом *Nit:*, чтобы автор знал, что это просто точка зрения, которую он может игнорировать.  
Примечание. Ничто в этом документе не оправдывает CL, которые определённо *ухудшают* общее качество кода системы. Такое возможно только в [чрезвычайной ситуации](https://google.github.io/eng-practices/review/emergencies.html).

**Задачи и цели проведения формальных инспекций.**

В тех случаях, когда верифицируется не программный код, а проектная документация на систему, которую нельзя "выполнить" или создать для нее отдельные тестовые примеры, также обычно прибегают к методу экспертных исследований программного кода или документации на корректность или непротиворечивость.

Такие экспертные исследования называют инспекциями или просмотрами. Существует два типа инспекций - неформальные и формальные.

При неформальной инспекции автор некоторого документа или части программной системы передает его эксперту, а тот, ознакомившись с документом, передает автору список замечаний, которые тот исправляет. Сам факт проведения инспекции и замечания нигде отдельно не сохраняются, состояние исправлений по замечаниям также нигде не отслеживается.

Формальная инспекция является четко управляемым процессом, структура которого обычно четко определяется соответствующим стандартом проекта. Таким образом, все формальные инспекции имеют одинаковую структуру и одинаковые выходные документы, которые затем используются при разработке.

Факт начала формальной инспекции четко фиксируется в общей базе данных проекта. Также фиксируются документы, подвергаемые инспекции, и списки замечаний, отслеживаются внесенные по замечаниям изменения. Этим формальная инспекция похожа на автоматизированное тестирование: списки замечаний имеют много общего с отчетами о выполнении тестовых примеров.

В ходе формальной инспекции группой специалистов осуществляется независимая проверка соответствия инспектируемых документов исходным документам. Независимость проверки обеспечивается тем, что она осуществляется инспекторами, не участвовавшими в разработке инспектируемого документа.

Входами процесса формальной инспекции являются инспектируемые документы и исходные документы, а выходами - материалы инспекции, включающие список обнаруженных несоответствий и решение об изменении статуса инспектируемых документов.

Этапы формальной инспекции и роли ее участников

Процесс формальной инспекции состоит из пяти фаз: инициализация, планирование, подготовка (экспертиза), обсуждение, завершение. В некоторых случаях подготовку и обсуждение целесообразно рассматривать не как последовательные этапы, а как параллельные подпроцессы. В частности, такая ситуация может сложиться при использовании автоматизированной системы поддержки проведения формальных инспекций.