АВТОМАТИЗИРОВАННЫЙ АССИСТЕНТ КОД-РЕВЬЮ НА ОСНОВЕ АНАЛИЗА ИЗМЕНЕНИЙ В PULL REQUEST

Поборцева У.С., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Жвакина А.В. – канд. техн. наук, доцент

Аннотация. В статье рассматриваются проблемы ручного код‑ревью в современных командах разработки и обосновывается необходимость автоматизации анализа изменений в Pull Request. На основе анализа предметной области, статистических данных и существующих инструментов предложена архитектура автоматизированного ассистента код‑ревью, использующего статический анализ и методы интеллектуальной обработки кода. Представлены ключевые функции системы и её область применения.

Рост сложности программных систем, увеличение объёмов кода и ускорение релизных циклов приводят к повышению нагрузки на разработчиков, выполняющих код‑ревью. Согласно исследованиям, ошибки, допущенные на этапе сбора требований и ревью, составляют до 60 % всех дефектов проекта, а на выполнение ревью уходит от 20 до 40 % рабочего времени разработчиков. Среднее время ожидания первого ответа ревьюера составляет 4–12 часов, а полный цикл ревью – от одного до трёх дней [1].

Ручной анализ кода подвержен человеческому фактору, требует высокой квалификации и значительных временных затрат. Это создаёт объективную необходимость внедрения автоматизированных инструментов, способных снизить трудоёмкость процесса и повысить качество программных решений.

Код‑ревью является ключевым этапом обеспечения качества программного обеспечения. Его цели включают выявление ошибок, повышение читаемости кода, соблюдение архитектурных требований и предотвращение накопления технического долга. Однако ручное ревью имеет ряд ограничений: высокая трудоёмкость, зависимость от квалификации ревьюера, повторяемость замечаний (например, замечания по стилю кода, форматированию и дублированию), перегрузка специалистов при большом количестве Pull Request, задержки, влияющие на скорость релизов.

Статистические исследования подтверждают, что значительная часть времени ревью уходит на рутинные операции, которые могут быть автоматизированы. Автоматизация позволяет сократить длительность ревью на 20–40 % и уменьшить количество пропущенных ошибок.

Для оценки текущего уровня автоматизации были рассмотрены три класса инструментов:

1 Статический анализатор. Статические анализаторы представляют собой инструменты, выполняющие детерминированный анализ исходного кода на основе заранее определённых правил. SonarQube является типичным представителем этого класса. Преимущества: широкий охват правил для различных языков, наличие метрик качества (дублирование, сложность, покрытие тестами) [2]. Недостатки: анализ выполняется на уровне кода, SonarQube анализирует изменения в PR, но его анализ остаётся статическим и не учитывает семантический контекст или логику изменений; нет выявления сложных логических ошибок, выходящих за рамки статических правил. Данный тип анализаторов не использует машинное обучение для интеллектуальной обработки кода.

2 Сервис автоматического ревью, интегрирующийся с GitHub. Примером такого сервиса может являться Codacy. Он хорошо справляется с рутинными проверками, автоматически запускает проверку для каждого Pull Request, его легко настроить, но данный сервис ограничен статическими правилами и генерирует много малозначимых замечаний из-за большого количества правил по стилю и отсутствия семантического анализа [3].

3 Основанной на искусственном интеллекте SAST‑решение, на примере Snyk Code. Данное решение эффективно выявляет уязвимости благодаря использованию ML-модели [4]. Однако инструмент сфокусирован на безопасности, а не на общем качестве кода. Также Snyk Code не предоставляет комплексных метрик качества, таких как дублирование, стиль кода и покрытие тестами, поэтому не может быть полноценной заменой код-ревью.

Сравнение показывает отсутствие решений, объединяющих контекстный анализ изменений, интеллектуальную обработку кода и глубокую интеграцию с процессом ревью.

Разрабатываемая система представляет собой автоматизированного ассистента код-ревью, интегрированного с GitHub и ориентированного на анализ изменений в Pull Request. Она предназначена для того, чтобы повысить качество ревью и снизить нагрузку на разработчиков за счёт сочетания статического анализа, интеллектуальных методов и автоматизированной обратной связи.

Область применения системы: команды разработки, использующие GitHub, проекты с высокой интенсивностью изменений, организации, где требуется стандартизация ревью, а также команды без выделенных ревьюеров.

Доступ к функциональным возможностям ассистента разграничивается в соответствии с ролями пользователей: ревьюеры получают рекомендации по улучшению кода, структурированные отчёты и выявленные замечания, а администраторы – инструменты мониторинга и настройки правил анализа.

На основе анализа существующих аналогов можно выделить ряд ключевых преимуществ разрабатываемой системы. Её отличительной особенностью является сочетание традиционного статического анализа с методами машинного обучения, что позволяет выявлять не только формальные нарушения, но и более сложные логические и архитектурные проблемы, недоступные большинству современных инструментов. Важным преимуществом является и контекстный анализ изменений.

Гибкая настройка правил анализа обеспечивает адаптацию инструмента под реальные процессы команды и корпоративные стандарты разработки, а глубокая интеграция с GitHub позволяет автоматически публиковать комментарии и формировать отчёты непосредственно в рамках процесса ревью. Это снижает нагрузку на опытных специалистов, освобождая их от рутинных операций и позволяя сосредоточиться на архитектурных и концептуальных аспектах разработки. Дополнительным преимуществом является уменьшение стоимости владения благодаря внутренней разработке, что обеспечивает независимость от внешних поставщиков и возможность оперативной адаптации системы под потребности проекта. В совокупности эти факторы повышают прозрачность и воспроизводимость процесса ревью, формируя единый стандарт качества и снижая влияние человеческого фактора.

Основными функциями системы являются извлечение коммитов и изменённых файлов непосредственно из Pull Request, выполнение статического анализа и проверок стиля кода, интеллектуальный анализ кода с использованием ML‑моделей, выявление ошибок, уязвимостей и архитектурных несоответствий, автоматическая публикация комментариев в Pull Request, формирование отчётов и метрик качества, а также логирование действий для последующего аудита.

Система снижает трудоёмкость ревью, уменьшает влияние человеческого фактора и обеспечивает единообразие подходов к контролю качества.

Анализ предметной области, статистических данных и существующих решений подтверждает актуальность разработки автоматизированного ассистента код‑ревью. Предложенная система сочетает статический анализ, интеллектуальные методы обработки кода и глубокую интеграцию с GitHub, что позволяет повысить качество программных решений, сократить трудозатраты и ускорить процесс разработки.

Использование ассистента обеспечивает единообразие подходов к ревью, уменьшает влияние человеческого фактора и способствует формированию прозрачных и воспроизводимых процессов контроля качества.

Список использованных источников:

1. Kudrjavets, G. Does code review speed matter for practitioners? / Kudrjavets, G., Rastogi, A // Empirical Software Engineering. — 2023. — Т. 29, № 1. — Ст. 7.

2. SonarQube Server [Электронный ресурс]. – Режим доступа: https://docs.sonarsource.com/sonarqube-server. – Дата доступа: 15.03.2026

3. Why Codacy is the #1 SonarQube alternative [Электронный ресурс]. – Режим доступа: https://www.codacy.com/comparison/codacy-vs-sonarqube. – Дата доступа: 15.03.2026

4. Find, prioritize, and auto-fix issues with dev-focused SAST solutions [Электронный ресурс]. – Режим доступа: https://www.codacy.com/comparison/codacy-vs-sonarqube. – Дата доступа: 17.03.2026