ADScAI
Applied Data Science & Artificial Intelligence Symposium

Context-Aware Code Review: Integrating Generative AI for Automated Pull Request Analysis

1st Priyatharshan Balachandran
Research and Development
Information System Associates
Colombo, Sri Lanka
bpriyatharshan@isa.ae

2nd Rimaz Fawzer
Research and Development
Information System Associates
Sharjah, United Arab Emirates
rfawzer@isa.ae

***Keywords*—Automated Code Review, Generative AI, Context Awareness, Agentic Review**

I. Introduction

Pull request reviews in software industry are vital for ensuring code quality. Traditional manual reviews offer valuable human insights but can be inefficient. They also struggle with the challenges posed by rapidly growing, complex codebases. On the other hand, many automated tools focus only on syntax and style. They do not account for the broader business context. This paper presents a context-aware PR review system that combines generative AI, transformer-based embeddings, vector databases, and git diff augmentation to bridge the gap between technical accuracy and business needs. The goal is to provide clear feedback on both code implementation and intent, addressing challenges in large domain-specific codebases.

II. Literature Review

Recent advancements in automated code review include traditional tools like SonarQube and ESLint, which focus on predefined rules, and newer AI-driven systems. The work of Wang et al. [1] was inspired by the open-source project "Sage: Chat with any codebase" solution [2] that leverages Marqo to generate embeddings, offering a more resource-efficient alternative to heavy LLM-based approaches. While projects like PR-Agent (now QODO Merge) [3] offer open-source PR reviews, they lack contextual awareness. There are availability of proprietary systems that provide this, but are often costly at scale. Our work integrates modern AI pattern recognition with business context modeling, delivering cost-effective, context-rich code reviews that align technical insights with business goals, avoiding high computational and financial costs.

III. Materials and Methods

Figure 1 illustrates our system's high-level architecture, highlighting the integration of source control, Knowledge Base, LLMs, and knowledge extraction components. This section delves into the technical details behind these components, explaining how they work together to deliver intelligent PR analyses.

A. Preparing the Knowledge Base

The foundation of our system is a robust approach to building and incrementally updating a comprehensive knowledge base (KB) across multiple repositories, reflecting changes in the upstream production branch. First, the repository manager clones repositories if they haven't been cloned before. Next, the entire codebase is scanned and parsed using the Python Tree-sitter language parser, which generates a syntax tree. The code is then divided into chunks of 800 tokens, with an overlap of 200 tokens between consecutive chunks. Then, Marqo converts these chunks into vector embeddings with **hf/e5-base-v2** and stores them in the Vespa DB wrapped within Marqo. This process is executed in a multi-threaded manner to boost efficiency. It runs periodically, with our File Manager handling file version metadata to update only modified files and purge deleted content from the KB.

B. PR Analysis Pipeline

The PR analysis pipeline is designed to provide comprehensive and context-rich feedback by integrating multiple review components. Our PR review process starts by validating the PR event webhook from Bitbucket and creating initial entries in our PostgreSQL database. This applies to both PR creations and update events. Next, the diff processor cleans the git diff by removing comments, imports, and deleted content as well as partitioning the diff into multiple parts. The whole processing is based on the PR size and the token limits of the LLM we are using, ensuring optimal coverage of the changes while staying within token constraints.
Once preprocessing is complete, three tasks run concurrently. The PR enrichment module iterates over the diff parts to generate a summary that covers the overall business flow, core architecture changes, database queries, configuration modifications, security updates, and any included unit tests. This enriched summary is then incorporated into the PR description. Simultaneously, a generic reviewer evaluates the

https://doi.org/10.31705/ADScAI.2025.53

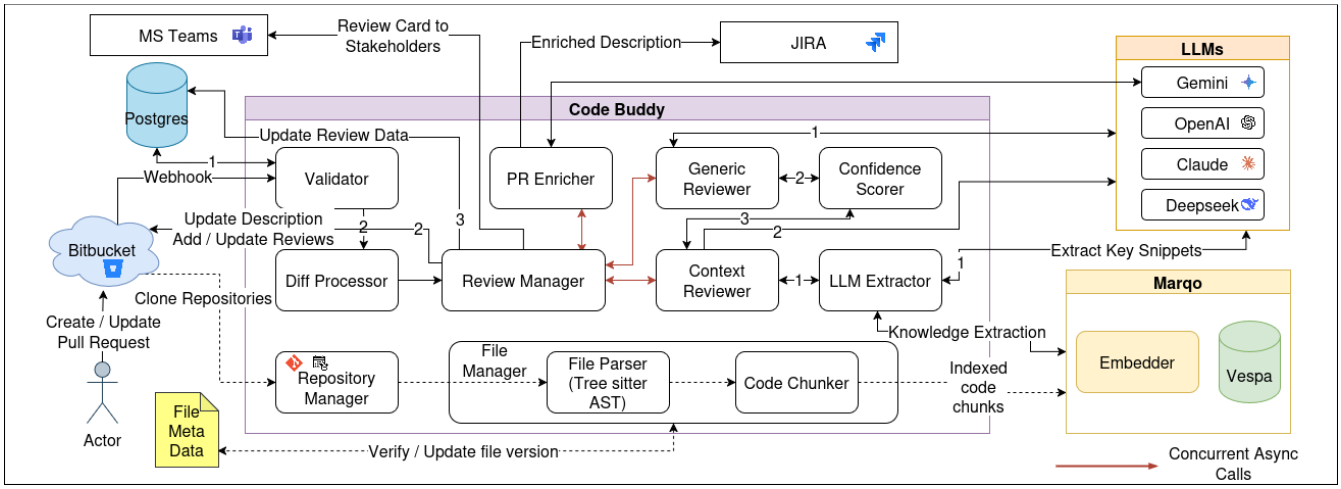

Fig. 1. High Level Architecture of the Automated Context Aware Reviewer

diff from a conventional coding perspective, focusing on standard practices and code quality. In parallel, the context reviewer leverages an LLM extractor to identify significant code snippets. It then performs a similarity search against our dedicated knowledge base to retrieve relevant information and pass it on to LLM in an agentic manner, providing additional insights on coding standards, file structuring, class usage, logic duplication, code reusability, and potential breaks in the business flow.

Following these reviews, a confidence scorer evaluates each comment against its corresponding code line, and only those comments that meet a predefined confidence threshold are passed on as review feedback. Structured output detailing the code line, review comment, and confidence score is produced via Pydantic models. Finally, notifications, including review content as MS Teams messages are sent to relevant stakeholders and JIRA ticket comments updates to give insights for QAs. Finally, the PR review metadata is updated in PostgreSQL. Here, we also handle the PR update events from Bitbucket as well as manage concurrent event requests for the same PR.

Our system features few technical additions. Highlightable feature is the line-number focused reviews that directly anchor feedback. For this we programatically augment the line numbers before passing to the LLMs. Incremental indexing to process only changed files, and efficient state management across multiple review rounds. Its LLM-neutral architecture supports various models (e.g., OpenAI, Gemini, DeepSeek), and a configuration-driven design lets teams set custom policies without altering the core.

IV. Results and Discussion

Our evaluations show noticeable improvements in review quality, efficiency, and consistency. Developers praised the system's contextual feedback, which captures not just what the code does, but why changes were made. This allowed reviewers to focus on architectural and logical issues rather than repetitive style-related comments. In many cases, context-aware analysis provided enough clarity for authors to adjust code before senior developers intervened, reducing redundant comments and improving review efficiency. New developers also found the feedback valuable for understanding project-specific coding standards as well as instances of functional duplication, accelerating onboarding. Teams reported smoother PR approvals with fewer redundant review cycles, leading to a more streamlined review process.

A. Future Work

Looking ahead, future improvements include expanding support across branches and product lines while integrating developer feedback for better review quality. Potential integrations like chat-with-code could enable on-the-fly discussions about reviews, while automated code assistance remains an area for further exploration. Enhancing security and performance analysis and optimizing the system for large PRs are also key directions for future development.

V. Conclusion

Our context-aware PR review system enhances code reviews by integrating generative AI with vector databases, offering feedback aligned with internal best practices and domain knowledge, unlike traditional linters. This allows senior engineers to focus on higher-level design decisions and complex problem-solving. As a result, organizations benefit from faster, more effective code reviews, reducing time spent on routine tasks and improving overall efficiency in software development.

References

[1] Z. Z. Wang, A. Asai, X. V. Yu, F. F. Xu, Y. Xie, G. Neubig, and D. Fried, "CodeRAG-Bench: Can Retrieval Augment Code Generation?," arXiv preprint arXiv:2406.14497, 2025. [Online]. Available: https://arxiv.org/abs/2406.14497.

[2] Storia-AI, "sage," GitHub repository. [Online]. Available: https://github.com/Storia-AI/sage.

[3] QODO-AI, "pr-agent," GitHub repository. [Online]. Available: https://github.com/qodo-ai/pr-agent.