Enhancing Developer Productivity through Hybrid Knowledge Graph–LLM Integration in Software Engineering Environments

M. Vaishaly[1]*, S. Vasanthapriyan[2], T.D. Rubasinghe[3]

[1]*Department of Computing and Information Systems, Faculty of Computing, Sabaragamuwa University of Sri Lanka, Belihuloya ,70140, Sri Lanka*
[2]*Department of Data Science, Faculty of Computing, Sabaragamuwa University of Sri Lanka, Belihuloya, 70140, Sri Lanka*
[3]*Department of Statistics and Computer Science, University of Peradeniya, Peradeniya, 20400, Sri Lanka*
**vmahendrarajah@std.appsc.sab.ac.lk*

Modern software repositories contain highly interconnected artifacts, including source code, documentation, issue reports, pull requests, commits, and dependency information, making repository exploration and software maintenance increasingly challenging. Although Large Language Models (LLMs) provide valuable assistance for software engineering tasks, they frequently lack repository-specific contextual understanding and may generate unsupported or hallucinated responses when reasoning across multiple software artifacts. This study proposes an adaptive hybrid Graph Retrieval-Augmented Generation (GraphRAG) framework that integrates a repository-specific knowledge graph with graph and vector retrieval to generate accurate, explainable, and evidence-grounded repository reasoning. The research addresses the questions of whether graph-grounded retrieval improves repository question answering, reduces hallucinations, supports multi-hop reasoning, and enhances repository exploration efficiency. A mixed-method evaluation is adopted using both a large-scale synthetic software engineering dataset and five real-world open-source repositories: Spring PetClinic, OpenMRS, eShopOnContainers, Apache Airflow, and MLflow. The synthetic dataset is programmatically generated to simulate realistic software development environments by creating interconnected repositories, developers, commits, issues, pull requests, documentation, dependencies, and software components with predefined semantic relationships. This controlled environment provides complete ground truth, enables reproducible experiments, supports scalability analysis, and facilitates systematic ablation studies, while the real repositories validate the framework under practical software engineering conditions. Repository artifacts are parsed using static code analysis and repository metadata extraction, transformed into a Neo4j Knowledge Graph, and integrated with adaptive GraphRAG retrieval. A benchmark comprising 300 manually designed repository reasoning questions is developed across entity retrieval, dependency analysis, issue reasoning, documentation retrieval, pull request analysis, and multi-hop reasoning. Ground-truth answers are generated through validated Cypher queries and manual verification against repository evidence. Experimental evaluation against Plain LLM, Vector RAG, Graph RAG, and the proposed adaptive GraphRAG framework demonstrates improvements with an Exact Match score of 0.842**,** Precision of 0.891**,** Recall of 0.876**,** F1-score of 0.883**,** ROUGE-L of 0.901**,** BERTScore of 0.928**,** a hallucination rate of 0.071**,** multi-hop reasoning accuracy of 0.864**,** and an average repository task completion time of 41.8 second**s**, outperforming the comparison methods. These findings demonstrate that adaptive graph-grounded retrieval provides more reliable, explainable, and scalable repository reasoning while establishing a reproducible evaluation framework for intelligent software engineering assistants.

Keywords: developer productivity, GraphRAG, large language models, software engineering, software repository reasoning