Multi-Agent Framework for Software Documentation

Srikanth Badavath
Department of Computer Science
Virginia Tech
Blacksburg, USA
bsrikanth@vt.edu

Neelesh Samptur
Department of Computer Science
Virginia Tech
Blacksburg, USA
nsamptur@vt.edu

***Abstract*—Documentation is considered the single source of truth that keeps the development, quality assurance and the product teams aligned with the organization's progress and future goals. The duration of a project is generally longer than the tenure of the employees at the organization. Documentation comes in handy when newer employees have to take up the responsibilities of their predecessors. Hence it is a crucial step in the software engineering lifecycle. Developers in the industry have demanding deadlines and it becomes difficult for them to maintain documentation. We try to solve this problem by using agents to develop 3 kinds of documentation- Release Notes, Knowledge Transfer Documentation(Person-centric and Feature-centric). Though we do not claim our documentation will eliminate the need for human intervention, we do believe our work will reduce the time taken to develop such documentation.**

***Index Terms*—Software Engineering, Documentation, Knowledge Transfer, Release Notes.**

I. Introduction

Software documentation, be it technical documentation like High-level design, Low-level design, Knowledge Transfer(KT) Documentation or Release Notes is often time consuming to generate and is sometimes considered secondary to a developer's job. Despite the monotonous nature of this task, documentation remains a critical asset of the company that tracks the company's progress in a well defined format. Software engineers responsible for the development of a certain piece of code are not always going to be associated with the same team or even organization. This makes it imperative to maintain documentation of the contributions of every team member to ensure smooth transition to their successors, taking up their responsibilities. It also serves as a single point of reference to resolve conflicts between multiple teams. (Development, Testing, Product teams etc.)

The rise of Generative Artificial Intelligence(Gen AI), Large Language Models(LLMs) in particular have changed the nature of most jobs, including software development. These LLMs are capable of generating code and text given a prompt or an input. LLMs have been widely adopted to do a wide variety of things to make life simpler. Though these LLMs are so powerful, they have one downside, their inability to interact with the outside world autonomously. This is where agents come in, agents are LLM wrappers that are much more capable of performing complex multi-step tasks without human intervention due to their ability to perform tool use and have their own memory. Agents also have the ability of interacting with external tools like Databases, APIs and can be integrated into workflows. We, in our work, plan on building a multi-agent framework that takes advantage of these benefits of agents to automate the process of generating three types of software documents- Release Notes(RN), Person-centric KT(PKT) and Feature-centric KT(FKT), using meta data from a Version Control System like Github.
The repository that the user wants to track must be available locally to avoid running into any issues with the Github API Limits.We are planning to use the fastapi repository for this study. The user is allowed to input a prompt from the prompt library, that provides a set of pre-defined prompts that are most optimal to work with our tool.

We propose 4 agents :-

- **Prompt Categorizer Agent** :- Categorizes the prompt from the user, based on the type of document requested (RN,PKT or FKT)
- **Ingestion Agent** :- For each type of document, different data must be fetched from the Git tools :-
    - RN- Commits and code diffs between two release note-tags
    - PKT- All commits involving the author, the user has requested for
    - FKT- All file names and commit messages, semantically related to the feature requested by the user

  Filtering must be applied at this stage to ignore all bot-generated commits
- **Summarizer Agent** :- Takes in the information from the Ingestion Agent and performs code summarization
- **Publishing Agent** :- Takes these summaries, generates documents and commits them into a directory in the same repository that we are tracking

Note that the directory that stores the documentation must be ignored by Git to ensure that these commits do not interfere with the commit tracking in the previous steps
Note:- The above dataset, architecture and tools are tentative and are subject to change. However, the high level objective will remain the same. We would also like to clarify that this tool is not production-ready and is a Proof Of Concept(POC) and might not be perfect.

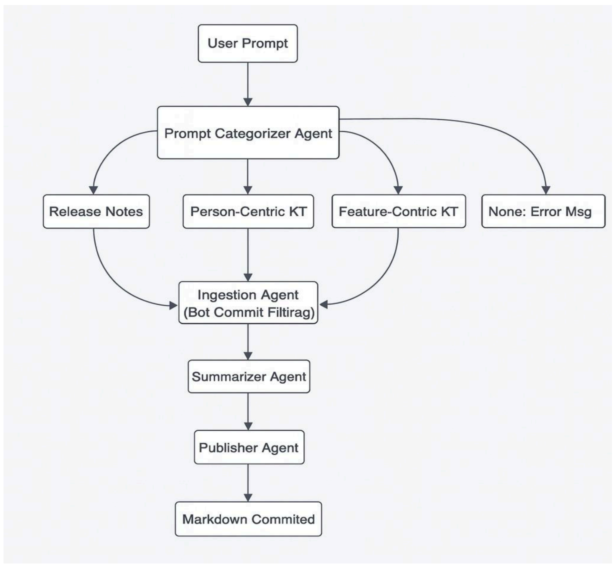

Figure 1: A high-level overview of our pipeline

II. Related work

Previous works in documentation automation include changelog generation, code summarization and knowledge transfer. For changelog generation, there are tools such as semantic-release or Conventional Commits, but they rely heavily on conference a specific commit format. There are also studies such as Chaturvedi et al. (2016) and Rahman et al. (2019) that extract release summaries using repository mining.

In the area of natural language summarization of code change, works by Buse & Weimer (2010) or Loyola et al.(2017) make contributions which are further improved by Angular, where transformer based models such as CodeT5 support documentation accuracy and contextual capturing. Furthermore, research by Storey & Zagalsky (2016) identifies structured KT as an important activity for onboarding into software development. Our contribution is related to the literature as we unify these various efforts through a multi-agent framework that automates the generation of both release notes and KT documentation in one end-to-end pipeline.

III. Schedule

Items	**Timeline (2025)**
Code setup, tool/library installation, understand git branching strategy of fastapi repo	**11th Oct - 12th Oct**
Learn LangGraph basics for multi-agent orchestration	**13th Oct - 19th Oct**
Build UI, Build Prompt Categorizer Agent	**20th Oct - 26th Oct**
Build Ingestion Agent	**27th Oct - 2nd Nov**
Build Summarizer Agent	**3rd Nov- 9th Nov**
Build Publisher Agent	**10th Nov- 16th Nov**
Integrate LangGraph Workflow, Test and Validate Pipeline	**17th Nov-23nd Nov**
Thanksgiving week	**23rd Nov- 30th Nov**
Final Report	**1st Dec - 7th Dec**

References

.
[1] C. Buse and W. Weimer, “Automatically documenting program changes,” Proc. IEEE/ACM Int. Conf. Automated Software Engineering (ASE), 2010, pp. 33–42.

[2] M. Loyola, M. Moreno, and A. Marcus, “Using concern graphs and NLP techniques to generate commit summaries automatically,” Proc. IEEE Int. Conf. Software Maintenance and Evolution (ICSME), 2017.

[3] M. Rahman, S. Roy, and C. Kästner, “Automatically extracting release notes from version control history,” Empirical Software Engineering, vol. 24, no. 3, pp. 1226–1257, 2019.

[4] T. Storey and A. Zagalsky, “Social media and software engineering: How developers collaborate and share knowledge through Twitter and GitHub,” Empirical Software Engineering, vol. 21, no. 2, pp. 1–32, 2016.

[5] S. Chaturvedi, M. Jain, and P. Saini, “Generating automated release summaries using issue and commit data,” Int. J. Adv. Comput. Sci. Appl. (IJACSA), vol. 7, no. 5, pp. 250–258, 2016.

[6] Y. Wang et al., “CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation,” Proc. Empirical Methods in Natural Language Processing (EMNLP), 2021.

[7] M. Karnin and A. D. Iyer, “Understanding developer turnover and its impact on software maintenance,” IEEE Trans. Software Engineering, vol. 47, no. 4, pp. 756–769, 2020.