Analyzing Developer-ChatGPT Conversations for Software Refactoring: An Exploratory Study

Soham Deo
sd5456@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Divya Hinge
dh8323@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Omkar Sandip Chavan
oc2464@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Yaxuan (Olivia) Wang
yxw1156@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Mohamed Wiem Mkaouer
mmkaouer@umich.edu
University of Michigan-Flint
Flint, Michigan, USA

ABSTRACT

In recent years, Large Language Models (LLMs) have witnessed a remarkable ascent, with OpenAI's ChatGPT, introduced in 2022, garnering substantial attention. ChatGPT's rapid adoption in the software development community has opened up new avenues for exploring its qualitative and quantitative impact on Developer-ChatGPT conversations. In this paper, we delve into a rich dataset from GitHub and Hacker News to perform a thorough analysis. Our objectives include characterizing the nature of these interactions and evaluating the use of ChatGPT in refactoring. To achieve these goals, we employ a combination of exploratory data analysis and data annotation, utilizing relevant keyword filters to extract pertinent information. Our examination encompasses the identification and analysis of code refactorings facilitated by ChatGPT. Through a meticulous exploration of these conversations, our goal is to illuminate the potential of ChatGPT to enhance software development practices. This research promises to provide valuable insights into the evolving role of ChatGPT in the world of software development.

CCS CONCEPTS

• **Software Engineering** → **Software Quality**; *Refactoring*.

KEYWORDS

Refactoring documentation, ChatGPT, mining software repositories

ACM Reference Format:
Soham Deo, Divya Hinge, Omkar Sandip Chavan, Yaxuan (Olivia) Wang, and Mohamed Wiem Mkaouer. 2024. Analyzing Developer-ChatGPT Conversations for Software Refactoring: An Exploratory Study. In *21st International Conference on Mining Software Repositories (MSR '24), April 15–16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3643991.3645082

MSR '24, April 15–16, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0587-8/24/04…$15.00
https://doi.org/10.1145/3643991.3645082

1 INTRODUCTION

The creation of new software projects and the maintenance of ongoing ones entail substantial efforts on the part of software developers. Over the past few decades, the development of Large Language Models (LLMs) has been on a constant rise [14]. The inception of OpenAI's ChatGPT in 2022 showcased multiple breakthroughs in various software engineering applications, in general, [2, 7–10, 13, 15], and refactoring, in particular, [1, 6, 11]. In addition, ChatGPT was introduced as a completely free service, which led to its user base skyrocketing in the past year[1]. Naturally, it has been used by developers to interact with, due to its quick and relevant responses when posed with questions related to software development and programming. The role of ChatGPT in software development has not been explored much due to its rapid adoption. In this study, we want to understand these conversations' qualitative and quantitative aspects.

The goal of this paper is to explore the nature of conversations between developers and ChatGPT and to further take a detailed look at the refactoring requests in these conversations. To this end, we answer the following Research Questions (RQs):

- **RQ1: What is the nature of Developer-ChatGPT conversations?** This RQ sheds light on the broader nature of conversations between developers and ChatGPT among the vast topics that can be discussed related to Software Engineering.
- **RQ2: How ChatGPT is used to perform code refactoring?** This RQ aims to check how developers perform refactorings using ChatGPT.
- **RQ3: On average, how many numbers of prompts are required to reach a conclusion ?** This RQ helps to understand the average conversation length between the developer and ChatGPT before coming to a conclusion.

2 EXPERIMENT DESIGN

Figure 1 shows the outline of our experiment design. In the following subsections we have described the dataset as well as experiment design steps in detail.

[1]ChatGPT sets the record for fastest-growing user base: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/

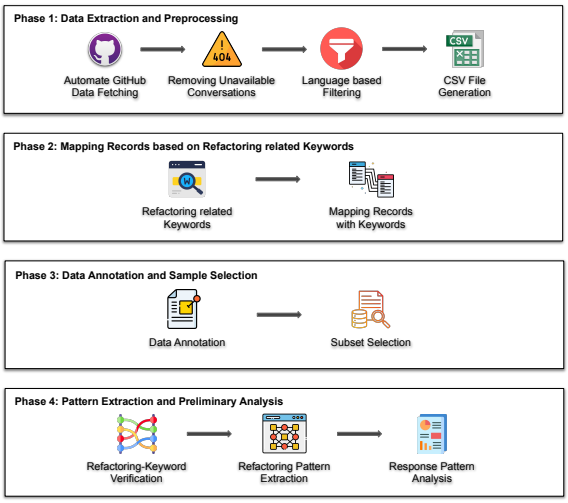

Figure 1: Experiment Design

2.1 Source Dataset

For our research, we have used the Dev-GPT dataset as presented by Tao X. et al [12]. This dataset is compiled from GitHub and Hacker News websites which encompasses data extracted from commit messages, issues, pull requests, and discussion pages across various public repositories. Specifically, information was collected by searching for ChatGPT sharing URLs on GitHub, as well as on the Hacker News forum. The organized dataset is categorized into directories, namely commit_sharings, discussion_sharings, pr_sharings, issue_sharings, and hn_sharings, with each folder containing relevant columns such as commit messages, titles, comments, URLs, and more. All of these directories consist of JSON files which represent snapshots captured on different dates. For this project, a snapshot generated on 2023-09-14 was selected.

2.2 Data Extraction and Preprocessing

To streamline our research, we developed Python scripts to automate the preprocessing of this dataset. This automation involved using GitHub API to retrieve the latest snapshots of files from the base dataset to be preprocessed. For preprocessing, we had two primary steps: Removal of ChatGPT URLs marked as unavailable (Error Status 404). Then, exclusion of records containing titles or messages in non-English languages. This was done using googletrans[2] package to detect the language present in the 'title' or 'message' column of each row in order to avoid processing through the individual prompts directly.

2.3 Mapping Records based on Refactoring Related Keywords

To identify prompts necessitating refactoring as requested by users or developers, we employed a regex pattern search. This method involved matching the presence of keywords in each prompt across all files and conversations. The patterns for the search were derived from the keywords provided Alomar et al. [3–5], with stems of each of the 29 keywords being individually searched. Subsequently, the identified keywords were appended to the 'keywords' column in the CSV file containing the conversations. This approach effectively filtered and categorized prompts where users or developers explicitly demanded refactoring.

2.4 Data Annotation and Preliminary Analysis

Data annotation was a pivotal component of our experiement design, involving annotation of conversations comprising of user prompts and ChatGPT responses across five distinct files: hn_sharings, issue_sharings, discussion_sharings, pr_sharings, and commit_sharings. Each file was carefully annotated by authors, with approximately 1500 prompt-answer responses per file.

The annotation process involved through analysis of user prompts and ChatGPT responses, wherein annotators provided the information related to prompt intent of a user, answer intent of ChatGPT, programming language used if any, the helpfulness of the suggestions by ChatGPT and any additional comments. Each conversation was carefully examined to understand its context.

Each author dedicated around 3 Weeks to complete the annotation of dataset. After that, authors compiled information about the prevalent topics discussed in the conversations, facilitating the classification of types of conversation such as documentation, issue, new feature, configuration, test, refactoring, and others. This annotation underwent cross-verification by the authors to rectify potential mislabeling of data.

2.5 Pattern Extraction and Preliminary Analysis

The annotated data underwent thorough analysis to verify whether the refactoring-related keywords used in prompts genuinely aligned with refactoring tasks. This examination provided valuable insights into discernible patterns associated with refactoring conversations. Additionally, a thorough examination of conversations not centered around refactoring was undertaken to reveal the broad spectrum of topics discussed with ChatGPT. Furthermore, a comprehensive analysis of the outputs produced by ChatGPT aided in discerning patterns within the responses.

3 EXPERIMENT RESULTS

We examined the conversations from the commit_sharings folder. After performing the extraction and preprocessing steps mentioned in Section 2.2 there were 1,554 records spanning 563 conversations. The duplicate conversations among these were filtered out, leaving 1,246 records of 447 conversations.

RQ1: What is the nature of Developer-ChatGPT conversations?

During the annotation of the data, it was observed that users had discussions with ChatGPT about various topics related to software engineering/development. These topics have been classified as follows:

[2]https://pypi.org/project/googletrans/

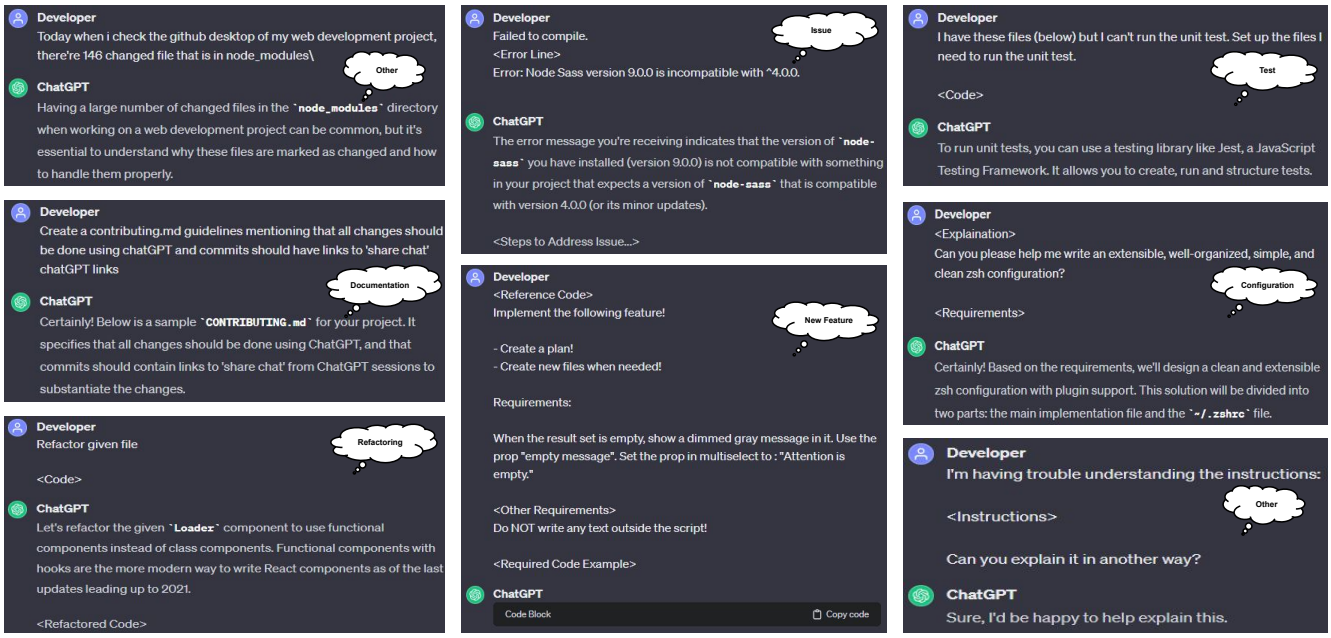

Figure 2: Conversation Topics with their associated labels

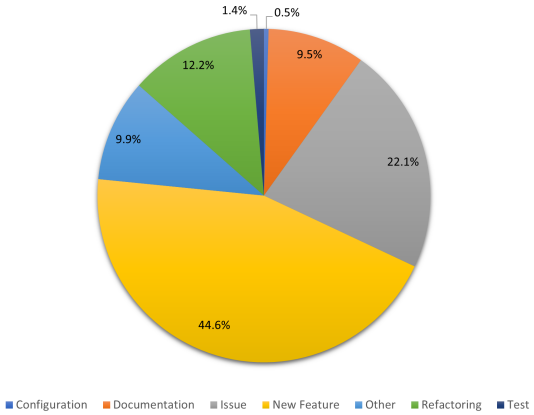

Figure 3: Topic Distribution

Documentation. Conversations related to documentation included, developers prompting ChatGPT to generate files with information related to project based on the requirement as well as improving documentation of code by adding appropriate comments. Nearly 9.5% of conversations were identified as documentation-related.
Example. Figure 2 shows an example of documentation where the developer is prompting the model to create a contribution.md file with specific guidelines.
Issue. In issue-related conversations, developers expected ChatGPT to resolve errors in code like syntax errors, compilation issues. The large number of conversations were focused on resolving code issues that contributed to the 22.1% of the conversations studied.
Example. Figure 2 shows an example of an issue where the developer is prompting the model to suggest solutions to an error message.
New Feature. In this type of conversations, developers provide requirements of what needs to be done to add a new feature. 44.6% of the conversations in the dataset were related to adding new features which were by far the most than all other conversation types.
Example. Figure 2 shows an example of a new feature where the developer is prompting the model to implement new features based on given requirements.
Configuration: These types of conversations were used to check whether the configurations of the server or the application are correct or not and to understand what can be improved.
Example. Figure 2 shows an example of a configuration where the developer is asking the model to design a clean and extensible *zsh* configuration with plugin support.
Test: Conversations related to testing were used to get testing scenarios from ChatGPT by providing the code functionality. A few of the conversations were also used to generate test cases for the provided code.
Example. Figure 2 shows an example of a test where the developer is prompting the model to run unit tests for provided files.
Refactoring: Refactoring-related conversations were used to optimize or improve code quality. A few of the conversations were also used to split the code to make it more readable and maintainable. 12.2% conversations focused on code refactoring.
Example. Figure 2 shows an example of refactoring where the developer is prompting the model to refactor a given file to use functional components instead of class components.

Other: All the conversations other than the above-mentioned categories were added to this category. 9.9% conversations were based on various types of discussions, which included working with GitHub, moving files from one directory to another, understanding instructions, asking ChatGPT to behave in a particular role by providing role details, and many more.
Example. Figure 2 shows an example of other where the developer is prompting the model to explain some instructions.

For these extracted topics, the distribution of their corresponding conversations can be seen in Figure 3.

RQ2: How ChatGPT is used to perform code refactoring?

The detailed analysis of the conversations shows that 54 out of 447 total conversations are related to refactoring. Majorly two contradictory patterns were observed in prompts: 1) The developer provides specific instructions of how the code should be refactored. 2) The prompt is open-ended, and ChatGPT suggests the outcome of refactoring like improving readability, maintainability, etc. For instance, In Figure 4 the developer provides specific instructions like performing refactoring by split and ChatGPT provides code accordingly without any suggestions. Whereas, in Figure 5, the developer lets ChatGPT suggest improvements, and ChatGPT specifies the refactoring outcome.

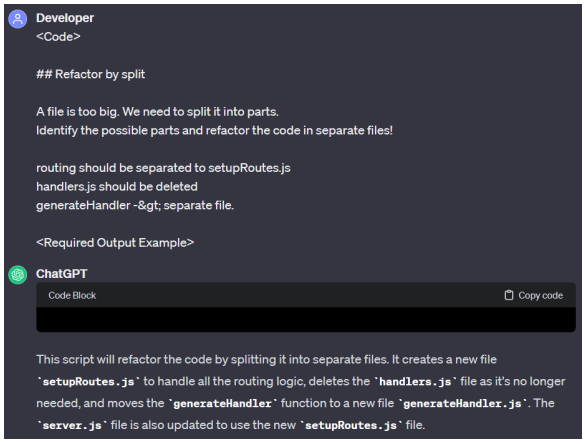

Figure 4: Refactoring Type 1

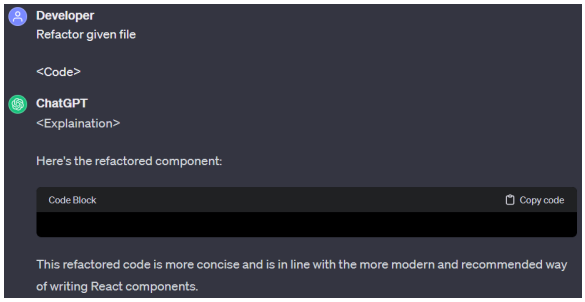

Figure 5: Refactoring Type 2

RQ3: On average, how many numbers of prompts are required to reach a conclusion ?

Table 1: Average Number of Prompts to reach Conclusion

Conversation Type	Average Number of Prompts
Commits	2.787472
Discussions	3.638298
Hacker News	4.062696
Issues	4.007958
Pull Requests	4.639344

Table 1 shows the average number of prompts provided by developer to reach the conclusion, if the suggestions provided by ChatGPT are useful or not. To get these numbers correctly we removed the duplicate conversation links from the dataset so that we can gather the accurate results. We observed that average number of prompts for conversations related to commits was around 3 which is least among all the other types of conversations, whereas discussions, hacker news and issues had average of around 4. The highest average number of prompts was for the pull request-related conversations which was around 5.

The above results help us to understand that as a developer, the changes to the source code are usually committed on a smaller scale but very frequently, whereas discussions, hacker news forums and issues take some time to reach the conclusion because there are various aspects which are required to be covered before coming to the conclusion. Pull Requests usually consist of a combination of many small changes to the code which require thorough review, which might be the reason these conversations take the highest number of prompts to reach a conclusion.

4 CONCLUSION

In conclusion, our study aims to investigate the impact of Large Language Models (LLMs) on software development, specifically focusing on the aspects in which developers rely on LLMs. Through the analysis of developer-ChatGPT interactions, we have attempted to identify patterns and discern the primary tasks within software development for which developers prefer leveraging suggestions from ChatGPT. Our findings reveal that developers predominantly seek guidance on topics such as code refactoring, documentation, issue resolution, error debugging, writing test cases, and integrating new features into existing code. Notably, within conversations related to code refactoring, two distinct prompt patterns emerged. One involved users providing step-by-step instructions, while the other featured users merely prompting ChatGPT to refactor the code. In the latter scenario, ChatGPT not only provided advanced solutions for refactoring but also articulated the anticipated outcomes, including improved code quality, enhanced maintainability, and reduced execution time. Furthermore, we analyzed the average number of prompts necessary to obtain a satisfactory response from ChatGPT. This aspect sheds light on the efficiency with which ChatGPT responds, without unduly consuming developers' time.

REFERENCES

[1] Aakash Ahmad, Muhammad Waseem, Peng Liang, Mahdi Fahmideh, Mst Shamima Aktar, and Tommi Mikkonen. 2023. Towards human-bot collaborative software architecting with chatgpt. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*. 279–285.

[2] Ali Al-Kaswan and Maliheh Izadi. 2023. The (ab) use of Open Source Code to Train Large Language Models. *arXiv preprint arXiv:2302.13681* (2023).

[3] Eman Abdullah AlOmar, Mohamed Wiem Mkaouer, and Ali Ouni. 2019. Can refactoring be self-affirmed? an exploratory study on how developers document their refactoring activities in commit messages. In *International Workshop on Refactoring-accepted. IEEE*.

[4] Eman Abdullah AlOmar, Mohamed Wiem Mkaouer, and Ali Ouni. 2021. Toward the automatic classification of self-affirmed refactoring. *Journal of Systems and Software* 171 (2021), 110821.

[5] Eman Abdullah AlOmar, Anthony Peruma, Mohamed Wiem Mkaouer, Christian Newman, Ali Ouni, and Marouane Kessentini. 2021. How We Refactor and How We Document It? On The Use of Supervised Machine Learning Algorithms to Classify Refactoring Documentation. *Expert Systems with Applications* 167 (2021), 114176.

[6] Eman Abdullah AlOmar, Anushkrishna Venkatakrishnan, Mohamed Wiem Mkaouer, Christian D. Newman, and Ali Ouni. 2024. How to Refactor this Code? An Exploratory Study on Developer-ChatGPT Refactoring Conversations. arXiv:2402.06013 [cs.SE]

[7] Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. 2023. Self-collaboration Code Generation via ChatGPT. *arXiv preprint arXiv:2304.07590* (2023).

[8] Wei Ma, Shangqing Liu, Wenhan Wang, Qiang Hu, Ye Liu, Cen Zhang, Liming Nie, and Yang Liu. 2023. The Scope of ChatGPT in Software Engineering: A Thorough Investigation. *arXiv preprint arXiv:2305.12138* (2023).

[9] Shuyin Ouyang, Jie M Zhang, Mark Harman, and Meng Wang. 2023. LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation. *arXiv preprint arXiv:2308.02828* (2023).

[10] Christoph Treude and Hideaki Hata. 2023. She Elicits Requirements and He Tests: Software Engineering Gender Bias in Large Language Models. *arXiv preprint arXiv:2303.10131* (2023).

[11] Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. *arXiv preprint arXiv:2303.07839* (2023).

[12] Tao Xiao, Christoph Treude, Hideaki Hata, and Kenichi Matsumoto. 2024. DevGPT: Studying Developer-ChatGPT Conversations. (2024).

[13] B Yetiştiren, Isik Özsoy, Miray Ayerdem, and Eray Tüzün. 2023. Evaluating the code quality of AI-assisted code generation tools: an empirical study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. arXiv preprint arXiv: 230410778. 2023.

[14] Ping Yu, Hua Xu, Xia Hu, and Chao Deng. 2023. Leveraging Generative AI and Large Language Models: A Comprehensive Roadmap for Healthcare Integration. In *Healthcare*, Vol. 11. MDPI, 2776.

[15] Jianzhang Zhang, Yiyang Chen, Nan Niu, and Chuang Liu. 2023. A Preliminary Evaluation of ChatGPT in Requirements Information Retrieval. *arXiv preprint arXiv:2304.12562* (2023).