RESEARCH-ARTICLE

Chatting with AI: Deciphering Developer Conversations with ChatGPT

SUAD MOHAMED, Belmont University, Nashville, TN, United States

ABDULLAH PARVIN, Belmont University, Nashville, TN, United States

ESTEBAN PARRA, Belmont University, Nashville, TN, United States

Published: 15 April 2024

MSR '24: 21st International Conference on Mining Software Repositories
April 15 - 16, 2024
Lisbon, Portugal

Conference Sponsors:
SIGSOFT

https://doi.org/10.1145/3643991.3645078
ISBN: 9798400705878

Chatting with AI: Deciphering Developer Conversations with ChatGPT

Suad Mohamed
Belmont University
Nashville, TN, USA
suad.mohamed@bruins.belmont.edu

Abdullah Parvin
Belmont University
Nashville, TN, USA
abdullah.parvin@bruins.belmont.edu

Esteban Parra
Belmont University
Nashville, TN, USA
esteban.parrarodriguez@belmont.edu

ABSTRACT

Large Language Models (LLMs) have been widely adopted and are becoming ubiquitous and integral to software development. However, we have little knowledge as to how these tools are being used by software developers beyond anecdotal evidence and word-of-mouth reports. In this work, we present a study toward understanding how developers engage with and utilize LLMs by reporting the results of an empirical study identifying patterns in the conversation that developers have with LLMs. We identified a total of 19 topics describing the purpose of the developers in their conversations with LLMs. Our findings reveal that developers use LLMs to facilitate various aspects of their software development processes (e.g., information-seeking about programming languages and frameworks and soliciting high-level design recommendations) to a similar extent to which they use them for non-development purposes such as writing assistance, general purpose queries, and conducting Turing tests to assess the intrinsic capabilities of the models. This work not only sheds light on the diverse applications of LLMs in software development but also underscores their emerging role as critical tools in enhancing developer productivity and creativity as we move closer to widespread AI-assisted software development.

CCS CONCEPTS

• **General and reference** → Empirical studies; • **Information systems** → *Data mining*; • **Computing methodologies** → **Artificial intelligence**; **Distributed artificial intelligence**; • **Software and its engineering**;

KEYWORDS

Large Language Models, LLM, chatGPT, Software Development, Empirical Study, Developer conversations

ACM Reference Format:
Suad Mohamed, Abdullah Parvin, and Esteban Parra. 2024. Chatting with AI: Deciphering Developer Conversations with ChatGPT. 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/10.1145/3643991.3645078

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

ACM ISBN 979-8-4007-0587-8/24/04
https://doi.org/10.1145/3643991.3645078

1 INTRODUCTION

The emergence of Large Language Models (LLMs) has offered a step towards revolutionizing problem-solving across several fields. Most notably, studies have found LLMs-based generative Artificial Intelligence (AI) models such as ChatGPT[1] to be diverse in the scope of their application throughout fields in medicine, education, and communication [8]. Software Engineering (SE) is one of the many fields in which LLMs are playing a crucial role. In recent years, many studies have aimed to analyze the scope of LLMs within the development sphere such as the ability to serve and perform software tasks [23]. On the other hand, many empirical studies suggest a gap in the limitations and capabilities of LLMs such as ChatGPT in its performance of comprehending developer tasks [9]. Despite the evident relevance, there is still a gap in our understanding of the usage and needs of developers regarding their relationship with LLMs beyond anecdotal evidence and word-of-mouth reports.

Previous studies have focused on identifying the breadth of how developers utilize LLMs through the characterization of systematic literature reviews and the extent of pre-processing and usage of software engineering-related datasets through LLMs [4]. We present an empirical analysis of conversations between software developers and ChatGPT to get a better understanding of the role that LLMs play in the software development process.

Our empirical study aims to provide insight into the range of LLMs applicability in the developer space. More specifically, we address the following research question: What are the most common usages or types of questions that developers present to ChatGPT? Our results show that developers use LLMs for a varied set of purposes. We found that developers' usages involve asking AI for help and assistance with both coding (57.02%) and non-coding tasks (e.g., writing, technical interview preparation, and general information queries) (42.98%). Furthermore, we found that the two most prevalent purposes in conversations between software developers and ChatGPT are 1) Developers asking AI to write specific code snippets (18.07%) and 2) general information queries for broad, non-coding related information (13.08%). The results of our study enhance our current understanding of the scope of LLMs in software engineering. Furthermore, we present preliminary results on leveraging machine learning to identify the purpose of the conversations automatically.

2 RELATED WORK

Machine Learning (ML) and AI for SE have demonstrated great potential to address software engineering problems[7, 11]. More recently, the introduction of transformer-based architectures has showcased the ability of transformer-based models to handle long-range dependencies thus becoming particularly effective in complex software engineering tasks [18].

Although LLMs are still in their early stages of development, tools such as GitHub Copilot[5], have demonstrated practical applications of LLMs in real-world coding scenarios, providing assistance in writing and suggesting code[17]. Recent studies have started examining the potential applications and limitations of AI assistants throughout the software development lifecycle and the shared roles they will have alongside human developers [4, 12, 13]. With the advancement of LLMs, studies have found multiple use cases where LLMs may improve the productivity of developers and excel at software engineering tasks such as specification generation, documentation, and language translation [13].

Several studies continue to look at the risks and constraints within the application of LLMs to software engineering tasks such as the reliability and security of code generated by AI assistants that utilize LLMs. [15]. Similar questions in differentiating content ownership with LLMs use have been highlighted showing the importance of maintaining a factor of human creativity [13]. Overall, software developers within the industry have varying opinions on the capabilities and use of LLMs in the development process. Nonetheless, research efforts have started exploring the use of generative AI models to address software engineering problems. Work by Xiao et al. [21] has already explored the use of LLMs for annotating data.

3 METHODOLOGY

The study utilizes the DevGPT dataset which comprises over 24,000 links to ChatGPT conversations with developers [20]. The dataset is divided into six subsets, or snapshots, all ChatGPT interactions were collected from sources such as GitHub or Hacker News at six specific time-frames that correspond to each snapshot. For this paper, the study surveys one of these six snapshots, snapshot 20230727, which consists of 1,929 links to ChatGPT conversations.

To identify patterns among the developer's conversations with ChatGPT, we performed an empirical analysis divided into two main steps. First, we performed a manual classification of the conversations using open coding to identify recurring conversation purposes. Second, using the conversation purposes from the manual classification, we explored a use scenario in which we leverage machine learning classification to obtain the purpose information.

3.1 Open Coding

To understand the purposes of the developers' conversations with AI, we employed a two-phase open coding approach toward identifying a comprehensive list of conversation purposes. Open coding is a qualitative categorization process that offers a method to define patterns and prevalent themes throughout a dataset[10, 16, 22].

The first phase was a trial phase in which two authors started with an empty set of categories and inspected a random set of 100 conversations from the dataset to derive a set of starting categories [10]. During this phase both annotators read the entire conversation, focusing on the prompts used by the developers and not the responses of LLMs. While reading each conversation, the authors asked the following question: "What is the purpose or goal of the developer?". They identified the purpose and the context of the prompts that developers present to ChatGPT and categorized them accordingly. If the identified category did not match an existing one, a new category was added to the list, and continued the labeling process with the new category list [16].

The results from the trial phase resulted in an initial list with 22 categories being identified. Each category was accompanied by a description that described the type of conversations that would fit within the category.

Following the trial phase, we selected a significant sample from the target snapshot with a 95% confidence interval and a 5% margin of error. This resulted in 321 conversations to be annotated during the coding phase which were not used during the trial phase.

During the coding phase, each annotator annotated the significant sample of conversations independently, using the 22 categories from the initial list with the option of creating new categories as needed if none of the existing categories fit the data. During the annotation process, any conversation from the dataset that was no longer available was labeled as 404 (not found) and any conversation not in English was labeled as N/A (not applicable). Therefore, the conversations labeled as N/A were not considered part of the sample as we were unable to decipher their contents. To account for this, the annotators evaluated a total of 412 conversations. Out of them, 91 were 404 or N/A leaving us with the 321 conversations analyzed.

First, two annotators independently annotated the 321 conversations using the label set derived during the trial phase, while creating new labels as needed. The resulting annotations from the coding phase showed a moderate agreement between the annotators with a Cohen's Kappa Agreement score of 0.584 [19]. Second, the annotators convened and collaboratively reviewed conflicting conversations by discussing the rationale for the annotations and refined the labels to be concise and representative. A total of 10 conversations could not be resolved by the annotators and required an additional mediator to resolve. After 15 hours of human effort, the finalized categorization resulted in 19 distinct categories, including two separate categories for links no longer available (404) and non-English conversations (N/A) (See Table 1).

3.2 Classification

Following the finalized annotation, we explored using our labeled dataset to evaluate a series of machine-learning classifiers to determine the usage categories automatically. We use the first relevant prompt presented by a developer as the input for the classifiers. This was done to ensure that any conversations that began with a greeting from a developer would not impact the results of the classifier.

The input developers' prompts were pre-processed with stopword removal and stemming. After pre-processing, the text was tokenized and vectorized using Term Frequency-Inverse Document Frequency (TF-IDF) before passing the encoded features and their corresponding labels to four different classifiers, Support Vector Machines (SVN), Naïve Bayes (NB), Random Forest, and Decision Tree. Each classifier was trained and tested on the vectorized sample of 321 conversations which excluded non-English and 404. We evaluated the performance of the classifiers using accuracy, precision, recall, and f1-score [2, 7, 14]. We used the NLTK library[1] in our experiments.

[1] https://www.nltk.org/

4 RESULTS

In this section, we divide our result into two parts. First, we discuss the results from our open coding annotation of the data and the categories that we identified. Second, we present preliminary results on the use of machine learning classifiers to derive the purpose information automatically.

4.1 Categories Identified

Our in-depth inspection and analysis of the conversations resulted in 19 distinct purposes, illustrating the diverse ways developers interact with LLMs. These interactions ranged from requesting the AI to write code(18.07%) and seeking clarification on software development frameworks(5.92%) to assistance with code comprehension(4.36%) and debugging(8.72%). The full list of identified purposes is shown in Table 1 a more detailed description can be found in our online appendix[2].

Table 1: Conversation Purposes Identified

Topics List	Occurrence
Write Code	58 (18.07%)
General Info	42 (13.08%)
Turing Test	34 (10.59%)
Simple Task	33 (10.28%)
Debugging	28 (8.72%)
How-to	25 (7.79%)
Framework/Dev tool inquiry	19 (5.92%)
Programming Language Inquiry	18 (5.61%)
Analyze Code	14 (4.36%)
Design Recommendation	12 (3.74%)
Writing Aid	8 (2.49%)
Customized Chatbot	5 (1.56%)
Operating Systems Inquiry	5 (1.56%)
ChatGPT Inquiry/Abilities	4 (1.25%)
Data Analysis	4 (1.25%)
Technical Interview Questions	4 (1.25%)
Troubleshooting	4 (1.25%)
Generate Visual/Text	3 (0.93%)
Generate Test Prompt	1 (0.31%)

Our most notable finding is that almost one-third of the conversations (31.15%) belong to the 'write code' and 'general information' categories, with 58 and 42 occurrences respectively. Furthermore, it is important to note that the second most frequent category, 'general information' is unrelated to software development. This shows that software developers who use ChatGPT acknowledge its value as a general tool to support them beyond software development. This is exemplified through the data, where we find that 41.43.% of the usages are for topics directly unrelated to software development. The third most common usage was Turing tests (10.59%), which were conversations in which the developers presented scenarios or asked logical questions to assess the logical comprehension capabilities of the LLM.

Conversely, when we focus exclusively on topics directly pertinent to software development, we see that 'debugging' was the second most common usage, accounting for 8.72% of the conversations. In these conversations, AI is predominantly used for identifying and fixing errors in specific pieces of source code. Moreover, conversations centered around code writing, debugging, and guidance on 'how-to' which implement certain code logic comprise 34.58% of the interactions between developers and ChatGPT. This highlights these purposes as key use cases and sheds light on the role that LLMs play in supporting developers throughout the software development process.

Finally, several categories within our dataset were marked by a relatively small number of occurrences, such as 'generate test prompt' and 'generate visual/text', represented by only 1 and 3 conversations, respectively. Nevertheless, we opted to retain these categories rather than merge them or remove them as they showcase the variability in how software developers use ChatGPT's extensive knowledge base and capabilities.

4.2 Representative Examples

Here we present and discuss a set of conversations to showcase the type of conversations that software developers have with conversational generative AI. First, Conversations A[3] and B[4] showcase the usage for code writing. Whereas, conversations C [5] and D[6], showcase their usage for non-coding purposes.

Conversation **A** ***(Write Code)*** involves a developer asking ChatGPT to write a Bash script to rename files and modify the files to follow snake_case notation instead of camelCase notation.

Conversation **B** ***(Write Code)*** shows a developer asking ChatGPT to write a Unity script in C# that would allow the shooting of projectiles using object pooling. The user described specific characteristics of the code, including classes and inheritance.

In both conversations, A and B, The developers indicated that the final code produced by ChatGPT was not exactly what the user expected but close enough to their desired goal.

Conversation **C** ***(General Info)*** shows a developer asking ChatGPT to provide information on what the Hacker News social website was and its demographics.

Conversation **D** ***(Turing Test)*** shows a developer performing a Turing test of ChatGPT by asking ChatGPT to create a new art form and make an artwork using this new form it has devised.

4.3 Classifiers preliminary results

Following the results of the manual classification, we explored an initial use case for the results from our study by exploring the automation of the classification process using various ML models including SVM, NB, Random Forest, and Decision Tree. In our experiments, the Random Forest classifier model performed with the highest accuracy with an F1 score of 0.582. To help better visualize the results across the categories we generated a confusion matrix (See Figure 1).

[2]https://figshare.com/s/f251dd154cd0119fd4d6

[3]https://chat.openai.com/share/7f2d011a-a8da-4fa4-9c6d-e09102fdd49f
[4]https://chat.openai.com/share/750e75a0-f7bc-4f4f-bb3e-1341713915d1
[5]https://chat.openai.com/share/66169853-d00a-4b05-abce-8f82c6d8c868and
[6]https://chat.openai.com/share/c2049bfe-7ab5-417b-9ceb-aca727d676c3

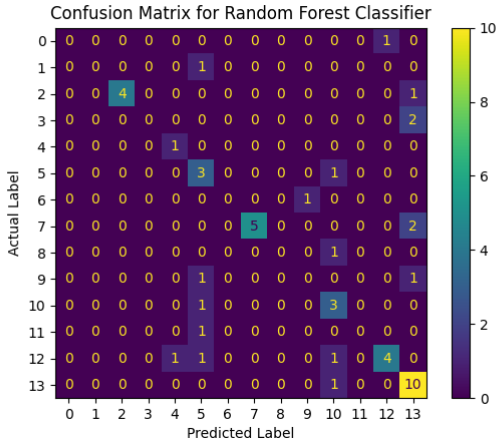

Figure 1: Confusion Matrix

After standardizing the ratio between the true and predicted values, we found the model to be better at classifying certain labels than others. In particular, 91% of the conversations labeled as 'simple task' in the testing subset of the data were identified correctly. However, there were still numerous false positives when labeling conversations as 'simple task'. This could be attributed to the imbalanced data collected from the sample or the methods in selecting the parameter for splitting the data for training and testing. Overall, given the number of available labels, the model performed fairly in classifying the conversations with moderately high accuracy and f1 score for certain machine learning classifiers. Further work with a larger sample size could be used to build off of this initial test.

5 ETHICAL IMPLICATIONS

The DevGPT dataset is published as open-source and the conversation links were made available by the users themselves. However, there are privacy concerns regarding the lack of direct consent from each person to have their conversations as part of the dataset [20]. To be conscientious of the users' privacy, our study did not rely on any personal identifiers that were present in the conversations and was based entirely on the technical content of the conversations. Furthermore, although the results of our study are meant to illuminate the way that developers interact with conversational AI tools when participating in software development, we are aware that the result of our analysis could potentially be misused to throttle or lock common usages by developers to implement predatory monetization to increase the revenue generated by LLMs.

6 THREATS TO VALIDITY

Despite our efforts to mitigate validity threats, our study has inherent limitations and potential biases of [3]. In particular, the use of open coding and the characteristics of our dataset, including its source and sample size, introduce risks during the open coding process. While we adhered to standard practices in open coding and endeavored to minimize biases in interpreting developers' prompts, there remains a possibility that we introduced a degree of bias to the results as we may have overlooked certain use cases in annotating the conversations due to them not being present in our sample. Lastly, we choose to generate our start list by labeling a fixed amount of samples rather than using a saturation approach which can potentially lead to a smaller set of starter categories. However, we found that our approach resulted in a starter set with an appropriate number of categories for a starter list [10] and any categories we might have missed would be identified during the second phase of the coding procedure.

Furthermore, our results are based on a single snapshot within the DevGPT dataset. Therefore, the results from our study may not generalize to all the interactions between developers and conversational generative AI models.

7 CONCLUSIONS

Our paper resulted in the identification of 19 topics that describe the most common purposes and goals of developers in their interactions with ChatGPT. Our results show that 57.02% of the conversation's purposes are related to coding with the most common purpose being to use AI for code generation (18.07%).

Our analysis shows that most of the conversations related to software development are focused on implementation and maintenance, as opposed to design, creativity, or planning. There may be a perceived notion that LLM models have not yet reached full competence in more abstract tasks in the software development pipeline and that there may be some apprehension towards the use of LLMs in software development and issues such as privacy must be resolved before we see full integration [13].

Alongside writing code, the results of this study also revealed that developers also commonly presented prompts outside of coding with the second most common purpose being obtaining general information about current news or scientific facts(13.08%). To an extent, these findings coincide with and build on previous research done on the sentiment software developers hold towards the role of AI and LLMs in the development process. A previous study found that the automation of tedious tasks and the gathering of information were the two main motives when adopting software development bots [6]. This is supported by the two main categories identified in our analysis.

Overall, our analysis of ChatGPT prompts led to constructive results in identifying the common uses of ChatGPT in software development. Our results can be used to inform the adoption of LLMs in software development, based on their actual usage by software developers. Here, we find a focus on simple code generation and a large knowledge base to be the most beneficial. As LLMs continue to advance, the scope of application in software engineering may broaden which may also expand the types of issues developers may commonly present as well as the range of problems they may present to LLMs in the near future. Furthermore, we identified less common use cases that serve to highlight the versatility and potential of conversational AI to cater to a wide array of developer needs, even in less frequently encountered scenarios.

ACKNOWLEDGMENTS

ChatGPT-4 was used for copy editing the text of this article.

REFERENCES

[1] Open AI. 2023. GPT-4 is openai's most advanced system, producing safer and more useful responses. https://openai.com/gpt-4

[2] Saqib Alam and Nianmin Yao. 2019. The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis. *Computational and Mathematical Organization Theory* 25 (2019), 319–335.

[3] Apostolos Ampatzoglou, Stamatia Bibi, Paris Avgeriou, Marijn Verbeek, and Alexander Chatzigeorgiou. 2019. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Information and Software Technology* 106 (2019), 201–230. https://doi.org/10.1016/j.infsof.2018.10.006

[4] Christian Bird, Denae Ford, Thomas Zimmermann, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk, and Idan Gazit. 2023. Taking Flight with Copilot: Early Insights and Opportunities of AI-Powered Pair-Programming Tools. *Queue* 20, 6 (jan 2023), 35–57. https://doi.org/10.1145/3582083

[5] GitHub CoPilot. 2023. Copilot, The world's most widely adopted AI developer tool. https://github.com/features/copilot

[6] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. 2020. An empirical study of bots in software development: characteristics and challenges from a practitioner's perspective. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'20)* (Virtual Event, USA). Association for Computing Machinery, New York, NY, USA, 445–455. https://doi.org/10.1145/3368089.3409680

[7] Abram Hindle, Earl T. Barr, Zhendong Su, Premkumar T. Devanbu, , and Mark Gabel. 2012. On the Naturalness of Software. In *International Conference on Software Engineering (ICSE-2012)* (Zurich, Switzerland). IEEE, 837–847. http://softwareprocess.ca/pubs/hindle2012ICSE.pdf

[8] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Lin Zhao, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. 2023. Summary of ChatGPT-Related research and perspective towards the future of large language models. *Meta-Radiology* 1, 2 (2023), 100017. https://doi.org/10.1016/j.metrad.2023.100017

[9] Wei Ma, Shangqing Liu, Wenhan Wang, Qiang Hu, Ye Liu, Cen Zhang, Liming Nie, and Yang Liu. 2023. ChatGPT: Understanding Code Syntax and Semantics. arXiv:2305.12138 [cs.SE]

[10] Matthew B Miles and A Michael Huberman. 1994. *Qualitative data analysis: An expanded sourcebook*. sage.

[11] Chris Mills, Jevgenija Pantiuchina, Esteban Parra, Gabriele Bavota, and Sonia Haiduc. 2018. Are Bug Reports Enough for Text Retrieval-Based Bug Localization?. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 381–392. https://doi.org/10.1109/ICSME.2018.00046

[12] Ipek Ozkaya. 2022. A Paradigm Shift in Automating Software Engineering Tasks: Bots. *IEEE Software* 39, 5 (2022), 4–8. https://doi.org/10.1109/MS.2022.3167801

[13] Ipek Ozkaya. 2023. Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. *IEEE Software* 40, 3 (2023), 4–8. https://doi.org/10.1109/MS.2023.3248401

[14] Esteban Parra, Mohammad Alahmadi, Ashley Ellis, and Sonia Haiduc. 2022. A comparative study and analysis of developer communications on Slack and Gitter. *Empirical Software Engineering* 27, 2 (2022), 40.

[15] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do Users Write More Insecure Code with AI Assistants?. In *in Proceedings of the 3oth ACM SIGSAC Conference on Computer and Communications Security (CCS'23)* (Copenhagen, Denmark). Association for Computing Machinery, New York, NY, USA, 2785–2799. https://doi.org/10.1145/3576915.3623157

[16] Hareem Sahar, Abram Hindle, and Cor-Paul Bezemer. 2021. How are issue reports discussed in Gitter chat rooms? *Journal of Systems and Software* 172 (2021), 110852. https://doi.org/10.1016/j.jss.2020.110852

[17] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *in Proceedings of the Conference on Human Hactors in Computing Systems (CHI'22)*. 1–7.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[19] Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam med* 37, 5 (2005), 360–363.

[20] Tao Xiao, Christoph Treude, Hideaki Hata, and Kenichi Matsumoto. 2024. DevGPT: Studying Developer-ChatGPT Conversations. In *Proceedings of the 21st IEEE International Conference on Mining Software Repositories (MSR'24)*.

[21] Ziang Xiao, Xingdi Yuan, Q Vera Liao, Rania Abdelghani, and Pierre-Yves Oudeyer. 2023. Supporting Qualitative Analysis with Large Language Models: Combining Codebook with GPT-3 for Deductive Coding. In *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces*. 75–78.

[22] Farida El Zanaty, Toshiki Hirao, Shane McIntosh, Akinori Ihara, and Kenichi Matsumoto. 2018. An empirical study of design discussions in code review. In *Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement*. 1–10.

[23] Zibin Zheng, Kaiwen Ning, Jiachi Chen, Yanlin Wang, Wenqing Chen, Lianghong Guo, and Weicheng Wang. 2023. Towards an understanding of large language models in software engineering tasks. *arXiv preprint arXiv:2308.11396* (2023).