On the Use of ChatGPT for Code Review

Do Developers Like Reviews By ChatGPT?

Miku Watanabe[1,2], Yutaro Kashiwa[1], Bin Lin[3], Toshiki Hirao[1], Ken'ichi Yamaguchi[2], Hajimu Iida[1]

[1]NAIST, Japan — [2]Nara College, NIT, Japan — [3]Radboud University, The Netherlands

ABSTRACT

Code review is a critical but time-consuming process for ensuring code quality in modern software engineering. To alleviate the effort of reviewing source code, recent studies have investigated the possibility of automating the review process. Moreover, tools based on large language models such as ChatGPT are playing an increasingly important role in this vision. Understanding how these tools are used during code review can provide valuable insights for code review automation.

This study investigates for what purposes developers use ChatGPT during code review and how developers react to the information and suggestions provided by ChatGPT. We manually analyze 229 review comments in 205 pull requests from 179 projects. We find that developers often use ChatGPT for outsourcing their work as frequently as asking for references. Moreover, we observe that only 30.7% of responses to the answers provided by ChatGPT are negative. We further analyze the reasons behind the negative reactions. Our results provide valuable insights for improving the effectiveness of LLMs in code reviews.

CCS CONCEPTS

• **Software and its engineering** → *Empirical software validation*; **Maintaining software**; **Software evolution**; *Software design engineering*; *Software development process management*.

KEYWORDS

Code Review, ChatGPT, Empirical Study

ACM Reference Format:

Miku Watanabe[1,2], Yutaro Kashiwa[1], Bin Lin[3], Toshiki Hirao[1], Ken'ichi Yamaguchi[2], Hajimu Iida[1]. 2024. On the Use of ChatGPT for Code Review: Do Developers Like Reviews By ChatGPT?. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3661167.3661183

1 INTRODUCTION

Code review is an important activity for modern software quality assurance [3]. Previous studies have demonstrated that code reviews improve software readability [4], maintainability [15], security [6], and design quality [21]. Moreover, code reviews can help identify defects [4, 20] and technical debts [14]. In recent years, code reviews are often performed online (*a.k.a. Modern Code Review*[24]), using web platforms such as Gerrit[1], ReviewBoard[2], and GitHub Pull Requests.[3] While providing significant benefits, code review is also a time-consuming process, which requires lots of effort from developers [5, 23, 31].

To facilitate the code review process, several studies have proposed techniques to automatically generate code reviews [11, 18, 27, 29, 31, 32]. One potential major player in this field is ChatGPT, whose recent emergence has attracted lots of attention from the software engineering community. ChatGPT has already exhibited outstanding abilities for code generation [28]. Intuitively, developers might wonder what role large language models (LLMs) such as ChatGPT can play in automatic code review generation. Indeed, a previous study has revealed that developers sometimes refer to ChatGPT conversations during code reviews [33]. Understanding how LLMs like ChatGPT are used and how developers react to the LLM-generated responses can provide valuable insights for code review automation and speed up the code review process.

In this study, we manually analyzed 229 review comments containing ChatGPT sharing links in pull requests. First, we retrieved and categorized the purposes of using ChatGPT in code reviews. Then, we examined whether developers are satisfied with the answers/suggestions generated by ChatGPT. For those cases where developers reacted negatively, we further inspected the reasons behind the unsatisfaction.

To the best of our knowledge, this is the first study to investigate how developers co-work with ChatGPT in code reviews. We believe that the insights we provide can serve as the foundation for future studies investigating LLM-based code reviews.

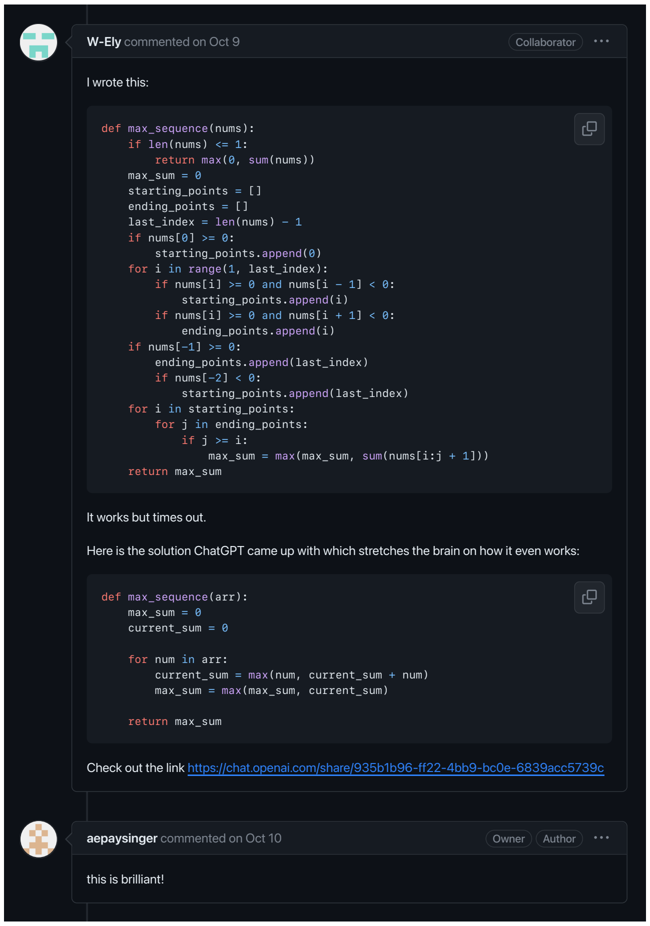
Replication. To facilitate replication and other future studies, we provide the annotated data and the used scripts on GitHub.[4]

2 MOTIVATING EXAMPLES

OpenAI has launched a new function that enables developers to share the prompts used by developers and the answers generated by ChatGPT since May 2023.[5] Thanks to this functionality, we could observe a lot of uses of ChatGPT during code reviews.

For example, in a pull request shown in Figure 1, a patch author ran across a performance-related issue and asked reviewers for ideas to resolve the timeout problem. To address this issue, one reviewer used ChatGPT, received a solution, and posted the link including the solution. The patch author also managed to understand the idea and liked it by saying "this is brilliant!". This example motivates us

[1]https://www.gerritcodereview.com
[2]https://www.reviewboard.org
[3]https://docs.github.com/en/pull-requests
[4]https://github.com/mmikuu/OnTheUseOfChatGPTForCodeReview
[5]https://help.openai.com/en/articles/7925741-chatgpt-shared-links-faq


W-Ely commented on Oct 9 · Collaborator

I wrote this:

```
def max_sequence(nums):
    if len(nums) <= 1:
        return max(0, sum(nums))
    max_sum = 0
    starting_points = []
    ending_points = []
    last_index = len(nums) - 1
    if nums[0] >= 0:
        starting_points.append(0)
    for i in range(1, last_index):
        if nums[i] >= 0 and nums[i - 1] < 0:
            starting_points.append(i)
        if nums[i] >= 0 and nums[i + 1] < 0:
            ending_points.append(i)
    if nums[-1] >= 0:
        ending_points.append(last_index)
        if nums[-2] < 0:
            starting_points.append(last_index)
    for i in starting_points:
        for j in ending_points:
            if j >= i:
                max_sum = max(max_sum, sum(nums[i:j + 1]))
    return max_sum
```

It works but times out.

Here is the solution ChatGPT came up with which stretches the brain on how it even works:

```
def max_sequence(arr):
    max_sum = 0
    current_sum = 0

    for num in arr:
        current_sum = max(num, current_sum + num)
        max_sum = max(max_sum, current_sum)

    return max_sum
```

Check out the link https://chat.openai.com/share/935b1b96-ff22-4bb9-bc0e-6839acc5739c

aepaysinger commented on Oct 10 · Owner · Author

this is brilliant!

Figure 1: Example of positive responses from patch-authors[6]

to investigate how reviewers can use ChatGPT for reviewing and we formulate our first research question as follows:

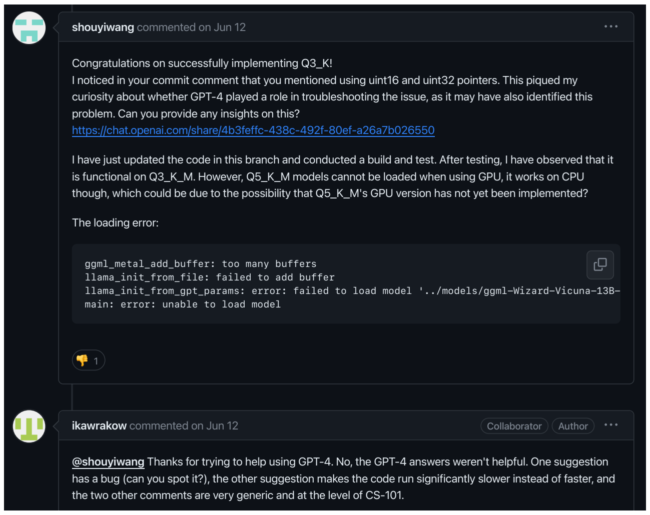
RQ$_1$***: For what purposes do developers use ChatGPT in code reviews?***

Unlike the aforementioned case, these reactions from patch-authors are not always positive. We observed one case where the patch author did not accept ChatGPT's suggestions. Figure 2 shows the example where a developer does not like the answer provided by ChatGPT. In the pull request, a patch author also encountered a performance issue and asked for help from reviewers. A reviewer consulted ChatGPT and presented two potential solutions provided by ChatGPT.

However, the patch author was not satisfied with these two solutions because one contained a defect and the other could lead to significant performance degradation. Motivated by this example, we posit our second research question:

RQ$_2$***: To what extent are developers satisfied with the answers provided by ChatGPT in code review related tasks?***

[6]https://github.com/aepaysinger/code-challenges/pull/65#issuecomment-1753175242
[7]https://github.com/ggerganov/llama.cpp/pull/1807#issuecomment-1586554971


shouyiwang commented on Jun 12

Congratulations on successfully implementing Q3_K!
I noticed in your commit comment that you mentioned using uint16 and uint32 pointers. This piqued my curiosity about whether GPT-4 played a role in troubleshooting the issue, as it may have also identified this problem. Can you provide any insights on this?
https://chat.openai.com/share/4b3feffc-438c-492f-80ef-a26a7b026550

I have just updated the code in this branch and conducted a build and test. After testing, I have observed that it is functional on Q3_K_M. However, Q5_K_M models cannot be loaded when using GPU, it works on CPU though, which could be due to the possibility that Q5_K_M's GPU version has not yet been implemented?

The loading error:

```
ggml_metal_add_buffer: too many buffers
llama_init_from_file: failed to add buffer
llama_init_from_gpt_params: error: failed to load model '../models/ggml-Wizard-Vicuna-13B-
main: error: unable to load model
```

👎 1

ikawrakow commented on Jun 12 · Collaborator · Author

@shouyiwang Thanks for trying to help using GPT-4. No, the GPT-4 answers weren't helpful. One suggestion has a bug (can you spot it?), the other suggestion makes the code run significantly slower instead of faster, and the two other comments are very generic and at the level of CS-101.

Figure 2: Example of negative responses from patch-authors[7]

3 STUDY DESIGN

3.1 Data Collection

Our study aims to understand how developers use ChatGPT during code reviews. More specifically, we would like to investigate what purposes developers use ChatGPT for (RQ1) and how developers react to the answers/solutions provided by ChatGPT (RQ2).

To reach our goal, we first need to identify the reviews in which developers use ChatGPT. To do so, we extend the dataset provided by Tao *et al.* [33]. The original dataset was created by identifying OpenAI's URL links to ChatGPT conversations (*i.e.,* links starting with *https://chat.openai.com/share/*) . They collected such links from files, commit messages, issues, pull requests, and discussions on GitHub. Here, our research mainly concerns the pull requests. While Tao *et al.* [33] provide several versions of the dataset, the latest version at the time of study dates back to October 2023, which does not have a sufficient amount of code review data. Therefore, we decided to collect new code review data in the same manner as the previous study using GitHub GraphQuery API.[3] We searched for the pull requests containing ChatGPT links from 27th May, 2023 (*i.e.,* the release date for ChatGPT share link) to 29th February, 2024.

We collected 509 pull requests (PRs) from 332 projects which contain links to ChatGPT conversations (606 review comments in total). However, in fact, these links can appear in both PR description and discussion comments, and not all of the comments are code reviews. To ensure that the studied ChatGPT uses are part of code review activities, we first excluded those links embedded in PR description and comments left by patch authors. This step led to 243 remaining review comments. We then manually went through all of them and removed invalid comments (*e.g.,* the deleted comments, comments not for reviewing, comments including the ChatGPT links cited from prior comments, etc.). Our final dataset includes 229 review comments. These review comments come from 205 pull requests collected from 179 projects.

[3]https://docs.github.com/en/graphql

3.2 Data Analysis

To answer RQ1, we performed a manual inspection to retrieve the purpose of using ChatGPT during code reviews. Two of the authors independently inspected each of the code reviews and annotated the purpose. In 75.2% of the inspected cases, the two authors reached an agreement, which led to a Cohen's kappa coefficient of 0.79, demonstrating a substantial agreement [16]. To resolve the disagreements, a third author checked the conflicting case and the corresponding annotations, and then recommended a final annotation. The recommendation was discussed with the first two authors until a full agreement was reached. The three annotators have 7 to 15 years of programming experience (7, 13, and 15 years, respectively).

To answer RQ2, we first categorized the attitude of the patch author toward answers/suggestions generated by ChatGPT. More specifically, we considered the following five categories: 1) positive, 2) negative, 3) neutral (often, the patch author praised the ChatGPT answers while pointing out some non-major issues), 4) no reply (the patch author did not reply by the time of data collection), and 5) ignored (the patch author replied but did not mention the ChatGPT answers). To better understand why patch authors are unsatisfied with the ChatGPT answers and provide actionable insights for future code review automation techniques, we manually analyzed the reason behind the negative attitudes. All the manual analyses conducted followed the protocol used to answer RQ1. As a result, we achieved 84.7% agreement rates with a Cohen's kappa coefficient of 0.71 (substantial agreement) for attitude categorization and 69.4% agreement rates with a coefficient of 0.68 (substantial agreement) for reason extraction [16]. Similarly, a third author resolved the conflicts throughout the discussions. Note that, in order to ensure accurate labeling, we decided to perform the manual inspection for classifying attitudes instead of using sentiment analysis tools [1, 7].

It is worth noting that there are several pull requests written in non-English languages. As the number of data points is not huge, to prevent further reducing the samples and keep the diversity, we used translation services to understand the contents.

4 RESULTS

4.1 RQ_1: Purposes of ChatGPT Uses

In total, we inspected 229 review comments. During the inspection, we excluded 10 review comments because the comments did not contain enough information to derive a reliable annotation. In the end, we classified the purposes extracted from 219 review comments into two categories (Reference and Outsource) and 15 sub-categories, which are illustrated in Figure 3. The numbers inside the circles on the top right corner of categories represent the number of cases we identified for each purpose category. Below we describe the details of these (sub)categories.

4.1.1 Reference. This category refers to those cases in which ChatGPT is not directly asked to address developers' issues. Instead, developers use ChatGPT to gain understanding and find evidence to support their opinions. 112 cases fall into this category, which can be further classified into 7 sub-categories: Refactoring, Implementation, Design, Non-Programming Tasks, Testing, Documentation, and Others.

Refactoring. Developers use ChatGPT to give references for better programming practices. The information provided might help improve code maintainability, performance, security, readability, etc. We found 35 such cases, accounting for 16.0%. For instance, a reviewer pointed out that the submitted JavaScript patches are too imperative, and asked ChatGPT to explain the differences between imperative and declarative programming with examples.[4]

Implementation. Developers use ChatGPT to learn certain development techniques. We observed 33 such cases (15.1%), which is the most common one in the "Reference" category. For example, a reviewer consulted ChatGPT on how to use the Callable type hint.[5]

Design. Developers consult ChatGPT to compare design choices. We observed 14 cases (6.4%) in this category. For example, a reviewer and a patch author discussed the approach for deleting records, and ChatGPT was asked to compare soft deletion and hard deletion.[6]

Non-Programming Tasks. Reviewers use ChatGPT to inform patch authors how to complete non-programming tasks (*e.g.*, environment setup). There are 10 such cases, which account for 4.6%. For example, a patch author inappropriately pushed commits and a reviewer used ChatGPT to explain how to rebase the commits.[7]

Testing. Developers utilize ChatGPT to demonstrate how to create test cases. We observed 7 cases of this category, which account for 3.2%. For example, a reviewer asked ChatGPT to create an example of using mockMVC in Spring.[8]

Documentation. Developers use ChatGPT to support their arguments regarding language issues in documentation. 6 cases (2.7%) fall into this category. For example, a reviewer asked a patch author to fix a grammar issue in the prompts used, by showing ChatGPT's answer as evidence.[9]

Others. 7 cases (3.2%) do not fall into the above categories.

4.1.2 Outsource. This category refers to those cases in which ChatGPT is directly asked to resolve problems or concerns of developers. There are 107 cases falling into this category, slightly less than the "Reference" category (112). These cases can be further classified into 8 sub-categories: Implementation, Refactoring, Bug-fix, Review, Testing, Design, Documentation, and Others.

Implementation. Developers directly ask ChatGPT to implement the program needed. 29 cases (13.2%) fit into this sub-category. Similar to the "Reference" category, "implementation" is the most prevalent use of ChatGPT during code reviews. For example, a reviewer asked ChatGPT for a solution to disable internet access during the GitHub Actions build process.[10] Another interesting example we found was that a reviewer tries to resolve Self-Admitted Technical Debt (SATD) [22] with the help of ChatGPT. In a review comment[11], a reviewer posted the link to the ChatGPT conversation in which he fed the SATD comment and the source code to ChatGPT and asked ChatGPT to generate implementations to address it.

[4]https://github.com/magnifiq/js-practice-vention/pull/9#discussion_r1353282723
[5]https://github.com/erobitschek/med-ml/pull/5#discussion_r1346318811
[6]https://github.com/GaloyMoney/galoy-mobile/pull/2361#issuecomment-1613522885
[7]https://github.com/Altinity/clickhouse-backup/pull/648#issuecomment-1592913792
[8]https://chat.openai.com/share/86b29c8a-ba4c-479c-9dfa-e102bc9c92a8
[9]https://github.com/huseyinbagator/react-demo-todo-app/pull/8#discussion_r1264647535
[10]https://github.com/erobitschek/med-ml/pull/5#discussion_r1346318811
[11]https://github.com/bancaditalia/black-it/pull/58#discussion_r1298585008

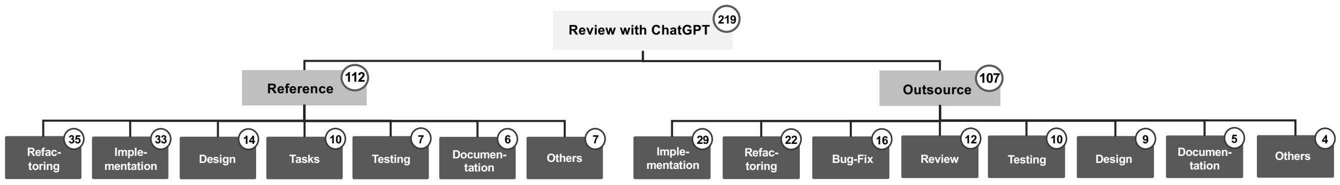


Figure 3: Classification results

Refactoring. In this category, developers ask ChatGPT to improve non-functional perspectives without modifying the behavior of the products. We observed 22 cases in this category, which accounts for 10.0%. For example, a reviewer asked ChatGPT to reduce the number of lines of code to improve readability.[12]

Bug-fix. Developers use ChatGPT to fix functional bugs. 16 cases (7.3%) fall into this category. For example, a reviewer asked ChatGPT how to fix an issue related to authorization flows by providing the descriptions of correct behaviors and actual behaviors.[13]

Review. Sometimes, developers ask ChatGPT to review the implemented code directly. We found 12 instances (5.5%) of this category. For example, a reviewer asked ChatGPT for the "thoughts on this code".[14] ChatGPT returned a few comments and one of them including a change example was merged into the repository.

Testing. Code reviews in this category employ ChatGPT to create test cases or verify the correctness of product behaviors. We observed 10 cases of this category, which account for 4.6%. For example, a reviewer asked ChatGPT to create a test method that verifies if a bug-fix patch is correct by giving the source code that does not work correctly.[15]Another interesting example is that a reviewer asked ChatGPT to verify whether two given implementations (one implemented with numpy library and the other implemented with `pytorch` library) have the same behavior.[16]

Design. In this category, developers ask ChatGPT to make design choices, often with rationales. We encountered 9 such cases, accounting for 4.1%. For example, a reviewer asked ChatGPT for the common practice of encouraging mobile app users who are satisfied with the app to leave a review.[17]

Documentation. Developers have also utilized ChatGPT to modify the documentation. We found 5 cases of this category, which account for 2.3%. For example, a reviewer simply sent ChatGPT the current documentation and asked ChatGPT how it can be improved.[18] ChatGPT listed several points for improvement and presented a sample revision.

Others. 4 cases do not fall into the above categories (1.8%).

Answer to RQ1: *Reviewers use ChatGPT to outsource implementation, refactoring, bug fixing, reviewing, and testing. Around half of the time, reviewers use ChatGPT to look for references.*

[12]https://github.com/tinygrad/tinygrad/pull/1661#issuecomment-1692144992
[13]https://github.com/transloadit/uppy/pull/4110#issuecomment-1709589496
[14]https://github.com/SSWConsulting/SSW.Website/pull/1284#discussion_r1303683136
[15]https://github.com/citusdata/activerecord-multi-tenant/pull/199#issuecomment-1575558178
[16]https://github.com/roboflow/supervision/pull/177#issuecomment-1643969353
[17]https://github.com/OpenAdaptAI/OpenAdapt/pull/228#issuecomment-1595122865
[18]https://github.com/konfuzio-ai/konfuzio-sdk/pull/296#discussion_r1243266447

4.2 RQ_2: Developers' Reactions to ChatGPT

We inspected the reactions to 229 review comments. Throughout the manual inspection, we found that 133 code reviews containing ChatGPT conversations received responses from patch authors, while 89 code reviews did not. For the remaining 7 code reviews, we were not able to determine because the comments did not contain enough information to derive a reliable annotation. Out of 133 responses, we excluded 7 responses because the reaction cannot be classified due to the lack of context and removed 19 responses that did not mention the ChatGPT-related discussions although they replied to the comments. In the end, there are 114 valid responses, including 73 positive responses, 35 negative responses, and 6 neutral responses. Most of the responses (64.0%) to ChatGPT's answers are positive while a non-negligible number of responses (30.7%) are negative. Interestingly, 65.7% of the negative responses (*i.e.,* 23 responses) are relevant to the "Outsource" categories in RQ1, which implies the necessity of careful checks by reviewers.

We then categorized the reasons for these 35 negative responses and extracted 12 reasons behind the negativity. Table 1 summarizes the reasons for negative responses and frequencies. The most common reason is that "The provided solution does not bring extra benefits" (7 cases). The second common reasons are "The provided solution contains a bug or is not working" and "The developer prefers another coding style/design choice" (6 cases each).

Answer to RQ2: *30.7% of responses to the ChatGPT's answers are negative reactions and the most common reason is "The provided solution does not bring extra benefits".*

5 FUTURE DIRECTION

In our study, we have revealed that developers tend to accept most of the review suggestions by ChatGPT. However, the negative reactions observed, especially when outsourcing reviews to ChatGPT, imply room for improvement. Studying why ChatGPT's suggestion does not reach developers' expectations would fill the gap between AI developers and practitioners. Our future work includes.

1. Developing automatic approaches for identifying the relevant reactions to new versions of ChatGPT: The evolution of LLMs is rapid and the collected reactions or the identified issues may no longer be valid, we will need to periodically collect and classify new reactions to monitor the effectiveness of LLMs in code reviews. One of the possible directions is to use sentiment analysis tools. The challenge here is to extract relevant review comments reacting to ChatGPT answers, due to the difficulty of untangling conversations mentioning different discussion points.

2. Developing context-aware code reviewing models: As a result of our study, we discovered that a few suggestions cannot

Table 1: Negative Categories

Reasons for negative comments	#
The provided solution does not bring extra benefits.	7
The provided solution contains bugs or is not working.	6
The developer prefers another coding style/design choice.	6
The developer has more optimal solutions for the issue.	4
The solution provided is generally true but is not for this case.	2
The solution needs more effort.	2
The query by the reviewer is inappropriate.	2
The provided solution does not consider specific restrictions/requirements of the project.	2
The provided solution contains bugs or is not working. Also, the provided solution degrades performance.	1
The solution provided is too general and overlooks details and specific features.	1
The developer is against the use of ChatGPT for certain purposes.	1
The reference is not needed anymore.	1

be accepted because they do not fit the context. For example, a reviewer admitted the suggestion is true in general cases but it does not work for the project.[19] One possible solution is to use the domain adaption technique that fine-tunes the tasks for a specific project. Fukumoto *et al.* [9] use this technique for code completion and demonstrate the ability of enforcing generated code to follow the coding rules of projects.

6 THREAT TO VALIDITY

Threats to internal validity: This study heavily relies on human annotations, which can be subjective. To mitigate the potential bias, we examined code reviews independently and discussed the review comments that have conflicting annotations until full agreement.
Threats to construct validity: This study utilized ChatGPT links to identify ChatGPT-supported code reviews. However, sometimes developers would directly use ChatGPT-generated solutions/suggestions without clear references to ChatGPT. The current dataset we use does not contain these cases, which might bias our results.
Threats to external validity: This study examines only 229 cases of code reviews using ChatGPT. These instances are all we could retrieve. One reason for such a small number is that the function of ChatGPT link sharing was only launched in May 2023. Replication studies are needed with extended datasets collected after a certain period of time to verify our results.

7 RELATED WORK

Emotions in Code Review: Egelman *et al.* [7] surveyed 1,317 developers to characterize the negative experiences and also proposed a model to identify five feelings of pushback during code review using log data. Ahmed *et al.* [1] have proposed a customized sentiment analysis tool for the software engineering tasks. Their approach achieved 83% accuracy in identifying negative review comments. Sarker *et al.* [26] developed an approach to detect toxic conversations in code review. Specifically, they fine-tuned a RoBERTa model with the 19,651 code review comments and found that the model achieved an F1-score of 0.88. Ferreira *et al.* [8] studied confrontational conflicts in code review discussions and found that over half (66.7%) of the non-technical emails associated with rejected changes included uncivil features in the Linux Kernel project.

These studies examine negative feelings in review conversations in general and do not focus on reactions against a specific context (*i.e.,* ChatGPT in this study).

Automated Code Reviews: In recent years, a lot of studies have employed generative models for code review. For instance, Tufano *et al.* [30] employed a Neural Machine Translation (NMT) model to translate submitted code to reviewed code. Similarly, Thongtanunam *et al.* [29] also automate code review tasks, using a Byte-Pair Encoding and a Transformer-based NMT architecture. Li *et al.* [18] evaluated a pre-trained model for four pre-training tasks tailored for code review, utilizing code changes and comments. Their evaluations show that the model outperforms other pre-trained models in terms of code change quality estimation, review comment generation, and code refinement. Qi *et al.* [10] compared ChatGPT with a state-of-the-art code review tool employing CodeT5 to examine the performance of the code refinement task. Their empirical results show that ChatGPT outperforms CodeReviewer.

These studies develop techniques to automate code reviewing while our work investigates *how* developers use generative approaches to support their code review tasks.

ChatGPT for Software Engineering Tasks: The role of ChatGPT in the realm of software engineering tasks has been particularly noteworthy, demonstrating the substantial impact. Sakura *et al.* [25] leverage ChatGPT and execution traces to identify the cause of bugs. Their experiment shows that execution traces can improve the performance of ChatGPT in identifying the cause of bugs. Li *et al.* [17] evaluated the applicability of ChatGPT for identifying failure-inducing test cases. They observe that ChatGPT can find correct failure-inducing test cases for buggy programs with a low success rate (28.8%). Eman *et al.* [2] examined refactoring activities with ChatGPT. They searched DevGPT [33] for discussions including refactoring-related keywords and disclosed the textual communication patterns of refactoring requests sent to ChatGPT. Jin *et al.* [12] examine how helpful the code generated by ChatGPT is. They show that 16.8% of the generated code is directly used, 26.0% is modified, and 32% is provided as supplementary information.

Several studies applied ChatGPT to software engineering tasks but reported negative results. Kabir *et al.* [13] studied ChatGPT's replies to 517 questions from Stack Overflow. They found that 52% of ChatGPT answers contain inaccuracies and 77% are verbose. However, participants preferred ChatGPT-generated answers 39%

[19]https://github.com/VOICEVOX/voicevox_engine/pull/904#discussion_r143335994

of the time. Liu *et al.* [19] identified and characterized potential issues with the quality of ChatGPT-generated code. They find that out of 4,066 programs generated by ChatGPT, 2,756 programs are regarded as correct, 1,082 programs return wrong outputs, and 177 programs contain compilation or runtime errors. Also, they observed that 1,930 code snippets suffer from maintainability issues.

These related studies have either focused on techniques for automating code reviews, or the application of ChatGPT in other SE tasks. There is currently no study to investigate the use of ChatGPT in code reviews, and we hope our study can fill this gap.

8 CONCLUSIONS

Our study involving 229 review comments from 179 projects explores how reviewers use ChatGPT for code review and to what extent patch authors like the code review suggestions provided by ChatGPT. As a result, we found that reviewers often use ChatGPT to outsource implementation, refactoring, reviewing, bug fixing, and testing, and they often use the suggestions by ChatGPT as evidence. In addition, we found that 30.7% of reactions to the ChatGPT's answers are negative and the most common reason is "The provided solution does not bring extra benefits".

ACKNOWLEDGMENTS

We gratefully acknowledge the financial support of JSPS for the KAKENHI grants (JP21H03416, JP24K02921), Bilateral Program grant JPJSBP120239929, and JST for the PRESTO grant JPMJPR22P3.

REFERENCES

[1] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. 2017. SentiCR: a customized sentiment analysis tool for code review interactions. In *Proc. of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE'17).* 106–111.

[2] 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. *CoRR* abs/2402.06013 (2024).

[3] Alberto Bacchelli and Christian Bird. 2013. Expectations, Outcomes, and Challenges of Modern Code Review. In *Proc. of the 2013 International Conference on Software Engineering (ICSE '13).* 712–721.

[4] Gabriele Bavota and Barbara Russo. 2015. Four eyes are better than two: On the impact of code reviews on software quality. In *Proc. of the 2015 International Conference on Software Maintenance and Evolution (ICSME'15).* 81–90.

[5] Amiangshu Bosu and Jeffrey C. Carver. 2013. Impact of Peer Code Review on Peer Impression Formation: A Survey. In *Proc. of the 2013 International Symposium on Empirical Software Engineering and Measurement (ESEM'13).* 133–142.

[6] Amiangshu Bosu, Jeffrey C. Carver, Munawar Hafiz, Patrick Hilley, and Derek Janni. 2014. Identifying the characteristics of vulnerable code changes: an empirical study. In *Proc. of the 22nd International Symposium on Foundations of Software Engineering (FSE'14).* 257–268.

[7] Carolyn D. Egelman, Emerson R. Murphy-Hill, Elizabeth Kammer, Margaret Morrow Hodges, Collin Green, Ciera Jaspan, and James Lin. 2020. Predicting developers' negative feelings about code review. In *Proc. of the 42nd International Conference on Software Engineering (ICSE'20).* 174–185.

[8] Isabella Ferreira, Jinghui Cheng, and Bram Adams. 2021. The "Shut the f**k up" Phenomenon: Characterizing Incivility in Open Source Code Review Discussions. *Proc. ACM Hum. Comput. Interact.* 5 (2021), 353:1–353:35.

[9] Daisuke Fukumoto, Yutaro Kashiwa, Toshiki Hirao, Kenji Fujiwara, and Hajimu Iida. 2023. An Empirical Investigation on the Performance of Domain Adaptation for T5 Code Completion. In *Proc. of the 30th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER'23).* 693–697.

[10] Qi Guo, Junming Cao, Xiaofei Xie, Shangqing Liu, Xiaohong Li, Bihuan Chen, and Xin Peng. 2024. Exploring the Potential of ChatGPT in Automated Code Refinement: An Empirical Study. In *Proc. of the 46th IEEE/ACM International Conference on Software Engineering (ICSE'24).* 34:1–34:13.

[11] Vincent J. Hellendoorn, Jason Tsay, Manisha Mukherjee, and Martin Hirzel. 2021. Towards automating code review at scale. In *Proc. of the 29th Symposium on the Foundations of Software Engineering (FSE'21).* 1479–1482.

[12] Kailun Jin, Chung-Yu Wang, Hung Viet Pham, and Hadi Hemmati. 2024. Can ChatGPT Support Developers? An Empirical Evaluation of Large Language Models for Code Generation. In *Proc. of the International Conference on Mining Software Repositories (MSR 2024).*

[13] Samia Kabir, David N. Udo-Imeh, Bonan Kou, and Tianyi Zhang. 2023. Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions. *CoRR* abs/2308.02312 (2023).

[14] Yutaro Kashiwa, Ryoma Nishikawa, Yasutaka Kamei, Masanari Kondo, Emad Shihab, Ryosuke Sato, and Naoyasu Ubayashi. 2022. An empirical study on self-admitted technical debt in modern code review. *Information and Software Technology (IST)* 146 (2022), 106855.

[15] Oleksii Kononenko, Olga Baysal, and Michael W. Godfrey. 2016. Code review quality: how developers see it. In *Proc. of the 38th International Conference on Software Engineering (ICSE'16).* 1028–1038.

[16] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33 (1977), 159–174.

[17] Tsz On Li, Wenxi Zong, Yibo Wang, Haoye Tian, Ying Wang, Shing-Chi Cheung, and Jeff Kramer. 2023. Nuances are the Key: Unlocking ChatGPT to Find Failure-Inducing Tests with Differential Prompting. In *Proc. of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE'23).* 14–26.

[18] Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, and Neel Sundaresan. 2022. Automating code review activities by large-scale pre-training. In *Proc. of the 30th Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE'22).* 1035–1047.

[19] Yue Liu, Thanh Le-Cong, Ratnadira Widyasari, Chakkrit Tantithamthavorn, Li Li, Xuan-Bach D. Le, and David Lo. 2024. Refining ChatGPT-Generated Code: Characterizing and Mitigating Code Quality Issues. *ACM Trans. Softw. Eng. Methodol.* (2024).

[20] Shane McIntosh, Yasutaka Kamei, Bram Adams, and Ahmed E. Hassan. 2014. The impact of code review coverage and code review participation on software quality: a case study of the qt, VTK, and ITK projects. In *Proc. of the 11th Working Conference on Mining Software Repositories (MSR'14).* 192–201.

[21] Matheus Paixão, Jens Krinke, DongGyun Han, Chaiyong Ragkhitwetsagul, and Mark Harman. 2021. The Impact of Code Review on Architectural Changes. *IEEE Trans. Software Eng.* 47, 5 (2021), 1041–1059.

[22] Aniket Potdar and Emad Shihab. 2014. An Exploratory Study on Self-Admitted Technical Debt. In *Proc. of the 30th International Conference on Software Maintenance and Evolution (ICSME'14).* 91–100.

[23] Peter C. Rigby and Christian Bird. 2013. Convergent contemporary software peer review practices. In *Proc. of the 2013 Joint Meeting of the European Software Engineering Conference (FSE'13).* 202–212.

[24] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern code review: a case study at google. In *Proc. of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP'18).* 181–190.

[25] Takafumi Sakura, Ryo Soga, Hideyuki Kanuka, Kazumasa Shimari, and Takashi Ishio. 2023. Leveraging Execution Trace with ChatGPT: A Case Study on Automated Fault Diagnosis. In *Proc. of the 39th IEEE International Conference on Software Maintenance and Evolution (ICSME 2023).* 397–402.

[26] Jaydeb Sarker, Sayma Sultana, Steven R. Wilson, and Amiangshu Bosu. 2023. ToxiSpanSE: An Explainable Toxicity Detection in Code Review Comments. In *Proc. of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'23).* 1–12.

[27] Shu-Ting Shi, Ming Li, David Lo, Ferdian Thung, and Xuan Huo. 2019. Automatic Code Review by Learning the Revision of Source Code. In *Proc. of the 33th AAAI Conference on Artificial Intelligence (AAAI'19).* 4910–4917.

[28] Dominik Sobania, Martin Briesch, Carol Hanna, and Justyna Petke. 2023. An Analysis of the Automatic Bug Fixing Performance of ChatGPT. In *Proc. of the 2023 International Workshop on Automated Program Repair (APR'23).* 23–30.

[29] Patanamon Thongtanunam, Chanathip Pornprasit, and Chakkrit Tantithamthavorn. 2022. AutoTransform: Automated Code Transformation to Support Modern Code Review Process. In *Proc. of the 44th International Conference on Software Engineering (ICSE'22).* 237–248.

[30] Michele Tufano, Jevgenija Pantiuchina, Cody Watson, Gabriele Bavota, and Denys Poshyvanyk. 2019. On learning meaningful code changes via neural machine translation. In *Proc. of the 41st International Conference on Software Engineering (ICSE'19).* 25–36.

[31] Rosalia Tufano, Simone Masiero, Antonio Mastropaolo, Luca Pascarella, Denys Poshyvanyk, and Gabriele Bavota. 2022. Using Pre-Trained Models to Boost Code Review Automation. In *Proc. of the 44th International Conference on Software Engineering (ICSE'22).*

[32] Rosalia Tufano, Luca Pascarella, Michele Tufano, Denys Poshyvanyk, and Gabriele Bavota. 2021. Towards Automating Code Review Activities. In *Proc. of the 43rd International Conference on Software Engineering (ICSE'21).* 163–174.

[33] Tao Xiao, Christoph Treude, Hideaki Hata, and Kenichi Matsumoto. 2024. DevGPT: Studying Developer-ChatGPT Conversations. In *Proc. of the International Conference on Mining Software Repositories (MSR 2024).*