

Tales From the Trenches

Expectations and Challenges From Practice for Code Review in the Generative AI Era

Nicole Davila^{ID}, Federal University of Rio Grande do Sul

Jorge Melegati^{ID}, Free University of Bozen-Bolzano

Igor Wiese^{ID}, Federal University of Technology

// In this study, we investigate what has been discussed about generative AI in the code review context by performing a gray literature review. We analyzed 42 documents and found insights from practice and proposals of solutions using generative AI models. //



THE EMERGENCE OF large language models initiated an era of generative AI, suggesting a disruption in many knowledge-based activities, including software development. In this era of generative AI-based solutions, where users receive recommendations from models trained on large databases, multiple efforts have been made to understand the potentialities and pitfalls of tools like ChatGPT and GitHub Copilot for software development.¹ Empirical evidence has shown promising results, especially regarding code generation.^{2,3} However, while several studies have explored the use of AI-based tools in code *writing*, more evidence still needs to be of adopting these tools to support the *review* of code changes.

Code review has been a well-established strategy in software engineering for quality assurance, relying on the manual effort and experience of developers (the reviewers) who check the quality of code changes proposed by a teammate (the author). In its contemporary variant, i.e., modern code review (MCR), the practice is characterized as frequent, tool-based, and flexible, being managed by tools such as Gerrit or the request mechanisms of Git-based systems.⁴ Several efforts have been made to improve MCR cost-benefit, including novel approaches and automation techniques.⁵ Such interest in tool support can leverage the capabilities of generative AI technology. For instance, code understanding has been challenging for reviewers, which motivates improvements in changes' visualization to ease code comprehension.⁴ However, a recent study has shown ChatGPT as an alternative to generating some sorts of explanations during MCR,⁶ which might mitigate understanding barriers. Therefore,

we might observe a shift in technologies used to build MCR solutions in the next years. A way to map potential future directions is to understand practitioners' perceptions.

This work investigates what has been discussed about generative AI-based tools in a code review context. To reach our goal, we perform a gray literature review (GLR). Previous work in the peer-reviewed scientific literature has summarized recurring themes and key findings from MCR studies up to 2021.^{4,5} However, generative AI-based tools' popularity has grown since 2022, not being covered by existing summaries. Due to the recent popularity and the existence of practitioner-produced content for and read by the software engineering community, we consider GLR a suitable approach. We analyzed 42 documents in our GLR and found insights from experience and proposals of solutions using generative AI models. Our study provides evidence on MCR practitioners exploring generative AI technology, especially ChatGPT.

What Characterizes Modern Code Review?

Nowadays, there is no single and widespread model for adopting code

review in software development. Empirical evidence has shown that multiple aspects influence MCR adoption, such as social and cultural factors, company policies, and available tool support.⁷ Nevertheless, based on ex-

interaction between reviewers and the author is through review comments written and presented by the supporting tool (Step 4). Depending on reviewers' feedback, new changes for repair or improvement may be

For instance, code understanding has been challenging for reviewers, which motivates improvements in changes' visualization to ease code comprehension.

isting work, we can identify a representation of commonly adopted MCR steps, as shown in Figure 1.⁴

When a developer finishes coding, she prepares the code change for review and includes a description of what was changed (Step 1), selecting appropriate reviewers (Step 2). Reviewers are notified, and upon accepting the code review invitation, they individually analyze the modified code for defects or opportunities for improvement (Step 3). The

necessary, and after being addressed by the author, a new iteration of code review may occur (Step 5). The process ends when reviewers decide whether to accept or reject the proposed changes (Step 6).

Therefore, given common aspects observed in real software development settings, modern code review can be characterized as a flexible, asynchronous, and tool-based practice, focusing on the quality of code changes.⁴

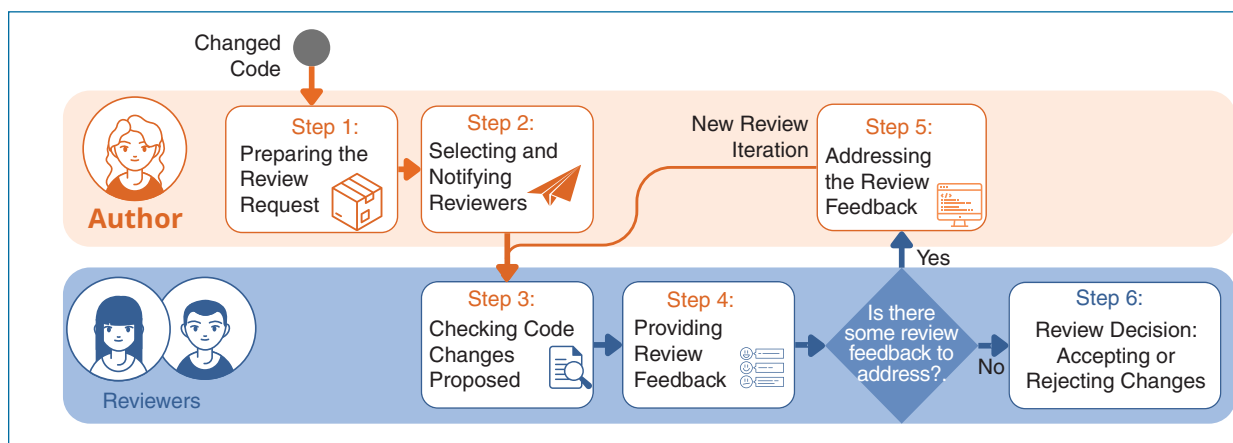


FIGURE 1. An overview of the steps of the modern code review process.

Methodology

We performed a GLR to address our goal following the guidelines proposed by Garousi et al.⁸ Figure 2 shows an overview of our research approach. Given space limitations, we provide more details about our study's procedure, including a list of selected documents, in the supplementary material at <https://doi.org/10.5281/zenodo.11216896>.

The document search phase focused on maximizing the return of text-based online documents on our study topic. We selected Google as the web search engine, and we tested several combinations of the search string regarding the amount of noise, i.e., the number of returned hits that did not match the scope of our study. We then selected the terms that produced less noise: “code review” and “generative AI,” followed by synonyms of the latter, i.e., “generative artificial intelligence,” “ChatGPT,” and “Copilot.” We searched on the international site of Google in October 2023 and obtained 382 results.

Next, we performed a selection phase (see Figure 2). We set

processing 191 results (50%) as our stopping criteria to make the process feasible. Later, we observed that most of our primary documents are from the top results retrieved by Google. Such observation suggests our primary documents are probably the ones practitioners have been considering. Given our subset of documents, we applied selection criteria that focus on text-based English publications produced by practitioners about MCR and generative AI. From 191 results assessed, 42 documents were selected for our study (22%). Almost three-quarters of our included documents (31/42, 73.8%) were among the first 100 search results.

In the data analysis phase (see Figure 2), coding and thematic analysis were conducted.⁹ We extracted excerpts and coded them using an inductive approach. While the first author coded the data, the other researchers checked the extraction for consistency, e.g., whether the code adequately represents a given data and whether groupings are coherent. We then explored the relationship between our codes and grouped

them into categories. For instance, codes referring to support reviewers in code understanding were considered related and grouped as “code change summarization.” All researchers compared and analyzed the resulting codebook regarding coherency, consistency, and minimal overlapping. As a result, we obtained key aspects discussed in the gray literature on generative AI and MCR.

As previously mentioned, details of our procedure are available in our supplementary material, given space limitations. Next, we discuss key results from our data analysis.

Tales From the Trenches

Practitioners from different areas of knowledge have explored tools based on generative AI for diverse purposes. In this study, we selected and analyzed 42 documents showing that code review practitioners have also explored such technology. Most of these documents (29 out of 42) are blog publications, e.g., LinkedIn posts. The remainder of the documents are available as company website publications (3), GitHub repositories (6), or a

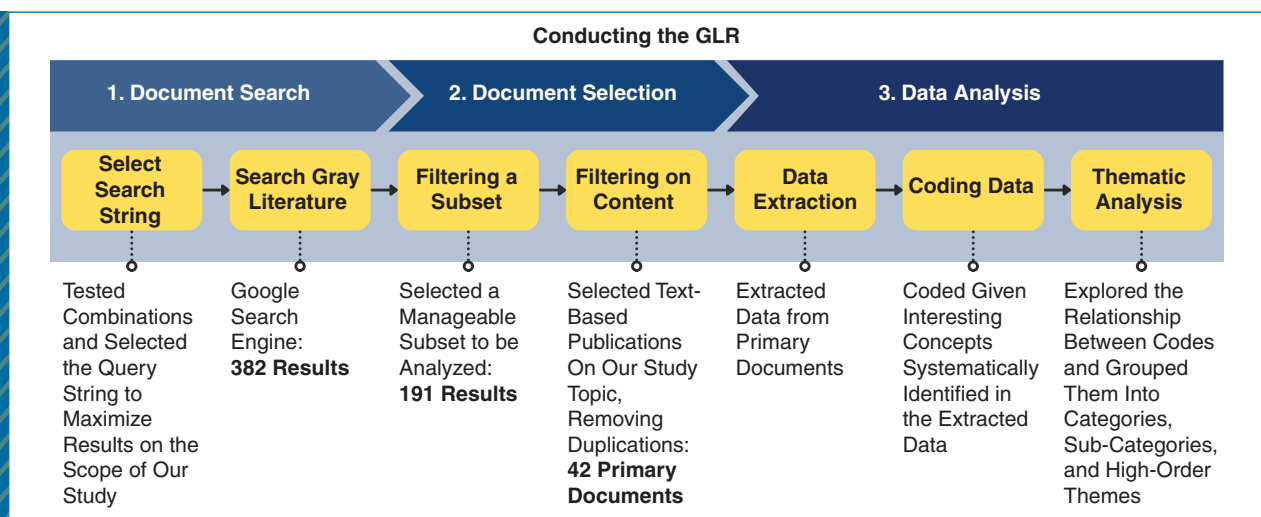


FIGURE 2. An overview of our research approach to conduct the GLR.

marketplace website (4). We provide a reference list of all the primary documents in our supplementary material.

We identified two main categories from our selected documents: *New Solutions* using generative AI to support MCR and *Insights from Practice* based on experiences with generative AI. In the former category, publications often express features and expected benefits when adopting the proposed tool in code review. In the latter, writers share their experiences, including suggestions for adoption and observed benefits when using existing generative AI-based tools for MCR.

After extracting the key intention (i.e., propose a new solution or share insights from practice) from the selected documents, we focused on perceptions of benefits related to using generative AI in the MCR context. Table 1 summarizes such expectations based on perceptions identified in our thematic analysis from GLR documents.

We did not identify documents on generative AI related to assisting neither the second nor the sixth step of the MCR process, i.e., selecting and notifying reviewers, and accepting and merging the code change (see Figure 1). One possible explanation

is the multiple contextual and human aspects often considered in code reviewer recommendation approaches and limitations for generative AI models do the same.

Next, we present the two main categories of documents identified in our GLR, i.e., new solutions and insights from practice, and discuss how each group relates to expectations summarized in Table 1.

New Solutions Using Generative AI for MCR

Of the 42 selected documents, 21 cases (50%) proposed new solutions.

Table 1. Perceived and expected benefits from GLR when performing MCR using generative AI-based tools. Documents organized into insights from practice (IFP) and new solutions (NS) may mention more than one expectation.

Expectation	Short description	IFP	NS
Step 1: Preparing the code review request			
Better Review Description	Support better writing of the descriptions often attached to code changes	-	5
Self-code Review Support	Support to review a code changes before code review	-	2
Step 3: Accepting the request and check the code			
Code Change Summarization	Facilitate change summarization to help understand it	3	5
Step 4: Providing feedback			
Review feedback provision	The generative AI technology can provide review feedback as part of the code review. This code includes cases in which the generation of review feedback is mentioned, but it is not specified whether it identifies coding issues or provides actionable fix suggestions.	17	17
Step 5: Addressing the review feedback			
Code Fix Execution Support	Support the developer in performing a code fix, repair, or code improvement suggested by the tool itself or human reviewers	2	1
Coding Issues Explanation	As part of the code review, the generative AI technology can provide explanations about a coding issue in a specific change to help the developer understand it.	2	-
General process benefits expected			
Better Cost-benefit	Reduce the process's overall duration, increasing the practice's efficiency	13	7
Early First Feedback	Provide faster first feedback on the code changes	2	2
Fair and Thorough Feedback	Provide review feedback comments that are fair and objective	2	1
Learning Support	Help practitioners learn about coding and best practices	1	1
Sparks Collaborative Discussions	Promote collaborative discussions during the code review process	2	1

Most of these proposals related to a novel reviewer bot (13/21, 60.9%). Often using GitHub Actions to trigger the bot, these proposals explore OpenAI's application programming interface to access the generative AI technology. Through open source project repositories hosted on GitHub, tool presentation pages, and insights for future development efforts, practitioners have been exploring the potential of this new technology as additional support to review a given set of code changes automatically.

Authors often highlight the available features and expected benefits when proposing a solution. Most cases mention support related to feedback provision (see Table 1): The generative AI-based tool will generate review comments on the code change. Although these documents often do not specify which aspects will be considered, some cases are more detailed, for instance, mentioning that the tool scrutinizes code changes line by line to detect coding problems related to logic errors,

violation of standards, antipatterns, and performance bottlenecks.

Insights from Practice

Another main category of documents identified in our GLR is Insights from Practice (21 of 42 documents). In this group, practitioners share adoption perceptions based on company studies and experience. Most frequently (19/21, 90.48%), these documents include a description suggesting a way to use existing generative AI-based tools for MCR. In these cases, the authors demonstrate familiarity with a tool, such as ChatGPT, and propose to the readers how to include it in the code review process. The focus is not on a new solution using existing technology but on adopting a tool as an assistant.

In this group, the publications usually highlight the benefits observed after experiencing generative AI-based tools. As in the new solutions, assistance in providing feedback is the most recurring benefit. However, in this case, the publications highlighted the potential to

flag coding issues and suggest actionable code fixes. Practitioners are more likely to indicate in this type of document the positive effect of using generative AI-based tools to reduce the process's overall duration, increasing the code review efficiency.

Challenges

A challenge represents a limitation when adopting generative AI in code reviews, and we identified five challenges, as shown in Table 2. Most challenges are not code review specific, referring to general difficulties related to using generative AI. Nevertheless, writers highlight these aspects as potential limitations that may negatively affect the review practice.

More often, publications alert about security and trustworthiness. When presenting the solution, the authors remind the user that the code change will be sent to another organization, e.g., OpenAI, encouraging a check on privacy and security policies. Furthermore, we observed warnings about not relying only on the proposed tool, as its suggestions

Table 2. Challenges when adopting generative AI in software development, including code review, are identified in the analyzed documents of our GLR. Documents organized into insights from practice (IFP) and new solutions (NS) may mention more than one challenge.

Challenge	Short description	IFP	NS
Generative AI Trustworthiness	Determining the trustworthiness of generative AI models without human intervention is often challenging due to the "black box" structure	3	1
Lack of Context	Rely on suggestions from generative AI models, given their limitations in understanding the context	3	-
Misleading Review Comments	Specifically rely on review comments from generative AI models, given the misleading or poor quality examples sometimes observed	2	2
Security and Privacy	Ensure the confidentiality of private code and compliance with security and privacy policies when using generative AI technologies	4	3
Token Limit	To provide large amounts of data given the current token limit of some generative AI-based tools. Processing large code review changes is difficult due to this limitation	1	2

may need to be more accurate or contain hallucinations.

Implications

The expectations when using tools based on generative AI can be associated with the code review process, as shown in Table 1 when referencing steps represented in Figure 1. While some implications from our findings might be expected, our results shed light on current discussions among practitioners and provide evidence to motivate future studies. We highlight three key implications observed in our GLR and discuss them next. Figure 3 also shows how they relate to the code review process presented in Figure 1.

Saving Time Support

Developer performance and cost reduction are recurring concerns motivating multiple efforts to create programming assistants. Several

studies in MCR research focus on saving time by proposing reviewer recommendation techniques and, more recently, automating code review tasks.^{10,11} Publications analyzed in this study suggest that tools such as ChatGPT and its base model can be alternatives for this purpose. For practitioners, by early flagging problems and how to solve them, this new family of tools can positively impact code review efficiency and reduce reviewers' workload. Future research can further explore to what extent these tools influence MCR outcomes (e.g., review duration), especially whether they are helping free reviewers focus on more subtle coding issues.

Cognitive Understanding Support

One of the challenges of code review is understanding the changeset,⁴ requiring the developer to process a large amount of information, including context and its implications, for code

comprehension and defect detection. This cognitive workload impacts reviewers' performance, motivating the proposal of visualization techniques and the definition of explicit review strategies to facilitate understanding a code change.^{4,12} Although not in-depth investigated, authors must also mentally process the reviewers' feedback and its implications. A recent study has shown evidence of ChatGPT as an alternative to generating some sorts of explanations during MCR,⁶ which might mitigate understanding barriers. An implication for researchers is to examine the impact of generative AI-based tools on code review practitioners' cognitive load and performance.

Reviewability Improvement

A prior study identified what makes reviewing a change more manageable.¹³ How well a change is explained, self-contained, and aligned with the project

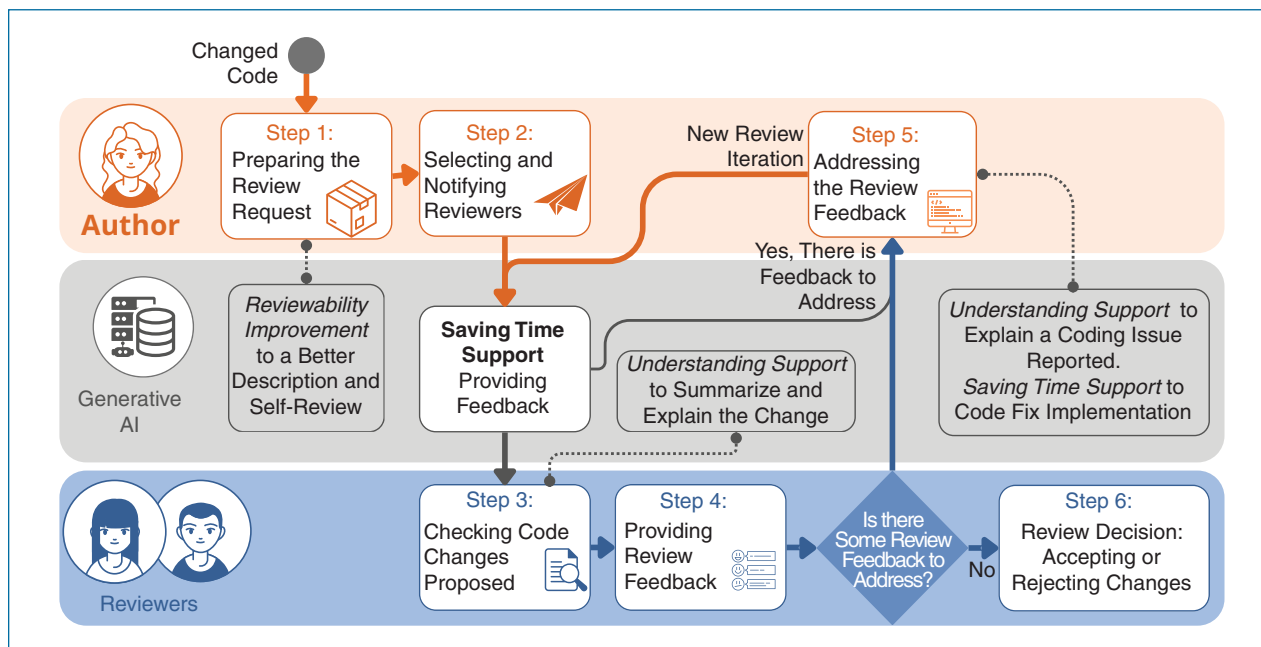


FIGURE 3. Given the steps of the modern code review process (Figure 1), potential implications from expectations related to generative AI-based tools.

style defines its reviewability, with the quality of the description being the most significant influence factor. Although present in few occurrences, evidence from our study suggests the potential that generative AI-based tools may have to assist in writing a better description, self-reviewing the code, or through review bots, positively impacting change's reviewability. To support such observation, a recent study examined how developers use Copilot in pull requests (PRs) and showed a growing adoption trend with positive effects on code review outcomes.¹⁴ For instance, using Copilot to write and review PRs reduced the review duration and increased the likelihood of acceptance. These results and our findings suggest the potential for generative AI-based tools to reviewability improvement.

Research Limitations

Generative AI and its applications have motivated multiple peer-reviewed scientific studies and publications by practitioners. Our review of the gray literature and its initial findings have limitations, which may influence the results presented in this article. We do not consider search results related to videos or forums, focusing on textual-based publications. Although these sources could be interesting, they would require much more resources to analyze, which is a reasonable criterion for considering or not a type of source in GLRs.⁸

Moreover, our search string is a potential threat that we mitigate by testing several combinations of keywords and selecting the one with less noise, i.e., with fewer returned hits that do not match the scope of our study. Nevertheless, this means other studies on generative AI and code review can present different perspectives and findings given their selection process

ABOUT THE AUTHORS



NICOLE DAVILA is a Ph.D. student at the Federal University of Rio Grande do Sul, Porto Alegre, 90040-060, Brazil. Her research interests include code review, software maintenance and evolution, and human aspects of software engineering. Davila received her master's degree in computer science from the Federal University of Rio Grande do Sul. Contact her at ncdavila@inf.ufrgs.br.



JORGE MELEGATI is a researcher at the Free University of Bozen-Bolzano, 39100 Bolzano, Italy. His research interests include human aspects of software engineering, software vulnerabilities, and agile software development. Melegati received his Ph.D. in computer science from the Free University of Bozen-Bolzano. Contact him at jorge.melegati@unibz.it.




IGOR WIESE is an associate professor in the Department of Computing, Federal University of Technology – Paraná, Campo Mourao, 87301350, Brazil. His research interests include mining software repositories techniques, human aspects of software engineering, and related topics. Wiese received his Ph.D. in computer science from the University of São Paulo. Contact him at igor@utfpr.edu.br.

decisions. Another limitation concerns the qualitative analysis, which one researcher mainly performed. As a mitigation strategy and to provide a sound classification of the documents analyzed, the other two researchers reviewed the data extraction and analysis, and all researchers participated in regular discussion meetings to mitigate biases and inconsistencies. The supplementary material provides more details about our study's procedure, list of selected documents, and analysis approach.

Our work's main contribution is the initial evidence of how generative AI-based

tools have been explored for code review in practice based on a GLR. We observed a high interest of practitioners in proposing new approaches that explore generative AI and suggesting adopting existing tools, such as ChatGPT. The most expected outcome is increased code review efficiency by automated feedback creation for code changes. In previous research, this acceleration mode of using tools like ChatGPT and GitHub Copilot has already been observed among software developers.¹⁵

Future research can further explore how practitioners interact with generative AI-based tools during code review and to what extent they have impacted the practice, especially as

support related to saving time, understanding changes, and reviewability of code changes. We also did not identify in our GLR discussions on generative AI related to assisting either the second or the sixth step of the MCR process, i.e., selecting and notifying reviewers and accepting and merging the code change. While decision-making support for reviewer selection is a popular topic,^{4,5} further exploring the adoption of this new technology for these steps is a potential avenue for research and industry advancement. We also observed a few code review-specific challenges related to generative AI being discussed, an aspect that can be explored by other studies. 

Acknowledgments

This study was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES)-Finance Code 001. Nicole Davila would like to thank CAPES for the research Grant 88887.480572/2020-00. Igor Wiese would like to thank CNPq/MCTI/FNDCT #408812/2021-4, and MCTIC/CGI/FAPESP #2021/06662-1.

References

1. A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. (J.) Jiang, "GitHub copilot AI pair programmer: Asset or liability?" *J. Syst. Softw.*, vol. 203, Sep. 2023, Art. no. 111734, doi: [10.1016/j.jss.2023.111734](https://doi.org/10.1016/j.jss.2023.111734).
2. B. Yetistiren, I. Ozsoy, and E. Tüzün, "Assessing the quality of GitHub copilot's code generation," in *Proc. 18th Int. Conf. Predictive Models Data Analytics Softw. Eng. (PROMISE)*, New York, NY, USA: ACM, Nov. 2022, pp. 62–71, doi: [10.1145/3558489.3559072](https://doi.org/10.1145/3558489.3559072).
3. N. Nguyen and S. Nadi, "An empirical evaluation of GitHub copilot's code suggestions," in *Proc. IEEE/ACM 19th Int. Conf. Mining Softw. Repositories (MSR)*, May 2022, pp. 1–5.
4. N. Davila and I. Nunes, "A systematic literature review and taxonomy of modern code review," *J. Syst. Softw.*, vol. 177, Jul. 2021, Art. no. 110951, doi: [10.1016/j.jss.2021.110951](https://doi.org/10.1016/j.jss.2021.110951).
5. D. Badampudi, M. Unterkalmsteiner, and R. Britto, "Modern code reviews - A survey of literature and practice," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 4, pp. 1–16, Feb. 2023, doi: [10.1145/3585004](https://doi.org/10.1145/3585004).
6. R. Widyasari, T. Zhang, A. Bouraffa, and e. D. Lo, "Explaining explanation: An empirical study on explanation in code reviews," 2023, *arXiv:2311.09020*.
7. T. Baum, O. Liskin, K. Niklas, and K. Schneider, "Factors influencing code review processes in industry," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, New York, NY, USA: Association for Computing Machinery, Nov. 2016, pp. 85–96, doi: [10.1145/2950290.2950323](https://doi.org/10.1145/2950290.2950323).
8. V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Inf. Softw. Technol.*, vol. 106, pp. 101–121, Feb. 2019, doi: [10.1016/j.infsof.2018.09.006](https://doi.org/10.1016/j.infsof.2018.09.006).
9. D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, Banff, AB, Canada. Piscataway, NJ, USA: IEEE Press, Sep. 2011, pp. 275–284, doi: [10.1109/ESEM.2011.36](https://doi.org/10.1109/ESEM.2011.36).
10. H. A. Çetin, E. Doğan, and E. Tüzün, "A review of code reviewer recommendation studies: Challenges and future directions," *Sci. Comput. Program.*, vol. 208, Aug. 2021, Art. no. 102652, doi: [10.1016/j.scico.2021.102652](https://doi.org/10.1016/j.scico.2021.102652).
11. R. Tufano, L. Pascarella, M. Tufano, D. Poshyvanyk, and G. Bavota, "Towards automating code review activities," in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE)*, Madrid, ES. Piscataway, NJ, USA: IEEE Press, May 2021, pp. 163–174, doi: [10.1109/ICSE43902.2021.00027](https://doi.org/10.1109/ICSE43902.2021.00027).
12. P. W. Gonçalves, E. Fregnan, T. Baum, K. Schneider, and A. Bacchelli, "Do explicit review strategies improve code review performance? Towards understanding the role of cognitive load," *Empirical Softw. Eng.*, vol. 27, no. 4, May 2022, Art. no. 99, doi: [10.1007/s10664-022-10123-8](https://doi.org/10.1007/s10664-022-10123-8).
13. A. Ram, A. A. Sawant, M. Castelluccio, and A. Bacchelli, "What makes a code change easier to review: An empirical investigation on code change reviewability," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Lake Buena Vista, FL, USA: ACM, Oct. 2018, pp. 201–212, doi: [10.1145/3236024.3236080](https://doi.org/10.1145/3236024.3236080).
14. T. Xiao, H. Hata, C. Treude, and e. K. Matsumoto, "Generative AI for pull request descriptions: Adoption, impact, and developer interventions," in *Proc. ACM Softw. Eng.*, 2024, pp. 1043–1065, doi: [10.1145/3643773](https://doi.org/10.1145/3643773).
15. S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA1, pp. 85–111, Apr. 2023, doi: [10.1145/3586030](https://doi.org/10.1145/3586030).