Leiden University

Informatica en Economie

A Systematic Literature Review of Generative AI for Code Review

Name:	David van Harrewijn
Student ID:	s3215059
Date:	15/07/2025
1st supervisor:	Joost Visser
2nd supervisor:	Miros Zohrehvand

BACHELOR'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Einsteinweg 55
2333 CC Leiden
The Netherlands

Abstract

Background: Generative AI (GenAI) is quickly being adopted in a broad range of software development tasks, including code review. SYMBALS is a technique for accelerating systematic literature reviews by applying AI for active learning, allowing the reviewers to more quickly select relevant papers.
Aim: We aim to conduct a systematic literature review of research on GenAI for code review, following the SYMBALS method. This allows us to uncover current research trends in applying GenAI to code reviews and to evaluate the SYMBALS method.
Method: To apply SYMBALS, we develop a protocol for our AI-assisted systematic literature review, including an provisional search query and criteria for including or excluding specific papers in the systematic literature review. This resulted in the final search query which we used in multiple databases to search for papers. We apply the ASReview tool to accelerate the selection process with active learning. Firstly we train the tool on which papers to include or exclude and then use the tool ourselves to include or exclude papers until the constraint has been reached. We then do the backward snowballing step manually to include the papers which had not been reached by the initial search query but are of importance. Finally, we analyse the identified literature by answering a small sets of research sub-questions for each article.
Results: We initially identified 541 papers and active learning allowed us to reduce this to a number of 47 relevant papers. To make this selection, we only needed to apply our selection criteria to 143 papers in the screening phase, due to active learning. At the end of the data extraction and synthesis step we eventually ended up with 22 papers. When evaluating the papers, we found that ChatGPT was the most often used GenAI technique for code review. The task where AI in code review was most often used was in the generation of comments for the code review. A small number of papers report some form of validation by the user or provide other information on whether the technique presented is ready for deployment. The SYMBALS method has proven to be effective for initial selection, but snowballing still needs a lot of effort.
Conclusion: SYMBALS is useful as a methodology, because of the amount of time which can be saved without presumably compromising on the quality of the paper. Research on GenAI for code review is still in its initial stages and is mostly theoretical and with few exemptions not applied in businesses or institutions.

Acknowledgements

Ik wil graag mijn scriptiebegeleider Joost bedanken voor alle feedback, motiverende gesprekken en zijn geduld met mij.

Ik wil graag Iris, René en Luna bedanken voor alle steun die zij tijdens mijn studie geleverd hebben.

Zonder jullie was dit echt niet mogelijk geweest!

Contents

Chapter 1

Introduction

Code review in the traditional form is a process between a reviewer and the developer, which can take up a lot of time. The question is could such a process already be made more efficient with generative AI. And if not, how far along in research is the field in this case. Since the introduction of OpenAI's ChatGPT model for the public in 2022, the amount and the variety of tasks where generative AI models are used have increased considerably. Generative AI is commonly praised for its usefulness in decreasing the amount of time needing to be spent on specific tasks. ChatGPT has already been used in a code review context, for instance by Yu et al. [55]. The question is what kind of research has been done on this subject and what are the current trends in the field of Gen AI application in code review.

To answer this question we use a systematic literature review, one of the most in depth ways to do research in a scientific field.

Since we expect the literature on GenAI-assisted code review to develop rapidly, it is important to speed up the literature review process. For this reason, we use the SYMBALS method [45] for systematic literature reviews that makes use of an active learning AI technique to reduce the effort needed from reviewers to evaluate literature for inclusion.

1.1 Research goals

For this research we have 2 goals. These goals in order of importance are:

- To find out the current trends within the application of Generative AI within code review.
- To find out if the SYMBALS methodology is a good option for a systematic literature review.

Below we explain which research question we formulated to help us reach these goals and we give a brief overview of our research approach.

1.2 Research question

The research question which this systematic literature review aims to answer is as follows:

RQ1 What are the current research trends in applying GenAI to code reviews?

1.3 Research approach

The approach of our research consists of using a systematic review to find out what the current trends are on the application of GenAI in code review. There are a lot of different ways of doing an systematic review, but we chose the SYMBALS methodology [45]. This methodology is a novel methodology published in 2021. We have chosen this methodology because a traditional systematic review was thought to be time consuming. SYMBALS has been shown to decrease the number of papers that need to be studied by humans from a full search to only a fraction even though presumably not decreasing the quality of the

papers in the review itself. The way SYMBALS decreases the number of papers that need to be studied by humans is that it makes use of AI in one part of its review, in particular its screening step.

1.4 Overview of the thesis

In Chapter 2, we discuss the concepts which are needed to understand the research question. In this chapter we discuss related papers as well. In Chapter 3 we discuss the methodology behind our research. In Chapter 4 we showcase the results which were found. In Chapter 5 we discuss the results, look at what kind of future research can be done on this subject and answer the sub-research questions. In Chapter 6, we wrap up the thesis with answering the main research question, mentioning the wider implications and the future work section.

Chapter 2

Background and related work

In this chapter, we discuss necessary background for our work as well as some related approaches.

2.1 Background

2.1.1 Systematic literature review

The purpose of a systematic literature review according to Schröer et al. [35], is to answer a research question based on distinguishing the relevant papers, assess those same papers and finally understand what those papers mean. Another paper gives a more direct definition, which is that the evidence out of those earlier mentioned relevant papers found by a specific "well-defined methodology" (Kitchenham et al. [20]) and unbiased way, will result in a systematic literature review, otherwise named, a secondary study to answer a specific research question [20].

2.1.2 SYMBALS

SYMBALS is a new methodology constructed by Van Haastrecht et al. [45]. It is a new systematic literature review methodology, that makes use of AI. Compared to the non-AI "state of the art", called FAST2 (Yu and Menzies [56]), SYMBALS outperformed it in speed and accuracy (Van Haastrecht et al. [45]). Despite their successful findings, this systematic literature review methodology has not been used often in the search database (Google Scholar, IEEE explore and ACM digital library) we consulted.

2.1.3 Code review

According to Thongtanua et al. Thongtanunam et al. [39], there is given a definition for software code review which is: "an inspection of a code change by an independent third-party developer in order to identify and fix defects before an integration". However the possibility of (semi-)automation of code review is not shown within this definition. These new techniques are for example "the Autotransform tool" which aids developers by automating "code transformations for code reviews" which reduces the amount of effort humans have to do for editing source code in a code review (Thongtanunam et al. [40]). Another example is GitHub's generative AI copilot for Pull Requests (PR) mentioned by Xiao et al. [54]. This same paper has seen an increased usage of the same generative AI method across different repositories.

2.2 Related work

One of the first papers where the SYMBALS methodology was applied is "a systematic review of socio-Technical cybersecurity metrics" Van Haastrecht et al. [46] by the same author that developed the methodology in the first place. This research was done, because of the lack of head-on generic systematic reviews on the topic of "socio-technical cybersecurity metrics" Van Haastrecht et al. [46] for the small and medium enterprises ("SMEs" Van Haastrecht et al. [46]). In the context of our research this systematic review was useful because it has a focus on computer science as well and this systematic review was done on a fairly large number of papers.

The second paper where the SYMBALS methodology was applied is a systematic review of "shared incident information"[44] to help improve cybersecurity by the same authors that developed the methodology in the first place Van Haastrecht et al. [44]. This paper is very similar in the sense of methodology, topic and authorship compared to the first one, but it has differences. Firstly this has a much more detailed focus than the more general one [46]. Secondly it has been shown to be smaller in scale. The first paper had 25773 papers during database search step of the Systematic literature review compared to 546 papers this iteration has. In the context of our research it is useful to showcase what the results would look like with a relatively similar number of papers. It has to be mentioned that the amount of information on how this systematic review was conducted is relatively low. It is for example, not mentioned what the exclusion criteria were or what the exact search query looked like. These two things are very important for conducting our research.

The third paper where the SYMBALS methodology was applied is a systematic review on "federated learning in mental health" Khalil et al. [19]. The third paper has used some useful metrics to showcase what the trends are in federated learning in mental health. Examples of these are the publishing years of each paper to showcase the trend of this research subject. Another example is to showcase from which publisher the paper is. In our opinion these metrics may by itself not be useful for directly answering a specific question but were useful to find out if the Systematic literature review was novel and how varied it is. This is important for our research because these metrics help with validating the results of the systematic literature review we end up with.

The final paper where the SYMBALS methodology was applied, is "a systematic review of Risk Stratification tools" [43] for ruling "out Acute Coronary Syndrome" [43]. when having the symptoms of chest pain at primary care van den Bulk et al. [43]. This paper is very interesting because it gives a view of how the use of the SYMBALS methodology, goes for a subject which is not computer science related. It is also a result dense paper, because every result is mentioned completely. Compared to the second paper [44] the full search query is mentioned and which database were used. Compared to the earlier mentioned paper, we find the lack of explanation of why certain choices were made not helpful. The paper gives no explanation on why only two databases were used. The paper also doesn't give any explanation on why certain search terms were used in the query. Compared to our research, this execution of the SYMBALS methodology lies the farthest away from our research. This is the case for the research field where the SYMBALS methodology is used on (medical versus computer science) and in how certain choices in the SYMBALS methodology are explained (almost none versus all of it). This paper is as well the hardest to understand for us, because it is written for people who have a background in the medical field, which for persons with a computer science background is hard to understand. Because this paper is result-dense we find it in our opinion a useful paper to see how the results of using SYMBALS in a systematic literature review can be written down.

While other papers use the SYMBALS methodology in full, some do not use the methodology in full, but only individual steps or other parts of the methodology. For instance, Eneche et al. [9] use only the backward snowballing step of SYMBALS in a literature review of "ground level temperatures in built up areas". In this paper a mix is used of the PRISMA protocol [30] with the SYMBALS approach. They found out that in that specific setup for the backward snowballing step, similar rates of recall occurred compared to systematic review completely using the SYMBALS methodology. This is interesting for our research, because it gives us an example of how to do the backward snowballing step.

In one of the papers the ASReview tool is explained by Quan et al. [32] in the form of a guide. The ASReview tool is used for the screening step in the SYMBALS methodology. The screening step in a literature review is used to find out what papers are relevant for answering the research question. It is the sole AI part of the methodology. This paper is very useful for our research, because it gives a tutorial in the form of an academic source for this relatively new model. Also, it provides an explanation that is easy to understand for someone who would have earlier never worked with such AI models.

Another paper which talks about ASReview but in a more critical light is Sijtsma et al. [37]. In this paper a practical application of the ASReview tool is discussed. This paper talks about the potential risks of using ASReview in a systematic review and how to mitigate them. This paper is very useful for our research because this allows us to mitigate the number of mistakes that could be made by using ASReview. For example, the results of the database search which ASReview will use, needs to be checked by a human for missing information. It could occur that no abstract or title comes with the export of the database search, two things which are essential for the screening process.

Chapter 3

Method

The methodology used to showcase the trends in the research field of application of generative AI to code reviews will be the SYMBALS method [45]. Each step of this SYMBALS methodology is explained here.

1. **Develop and evaluate protocol**
   We will use SYMBALS as a systematic literature review methodology. The first step in that method is setting up, and evaluating the protocol. The first step within that step is having a background, rationale and objective of the systematic review. Then a specific strategy is formulated for searching for all the papers with a query. From those papers we have to select which are relevant and which are irrelevant to train the AI tool [45].
2. **Database search**
   We find out what the consensus is on which databases are relevant for this discipline and are suitable for systematic literature review [45].
3. **Screening using active learning**
   We choose an AI tool with the ability to do screening with the given query and the possibility to change the stopping criteria [45].
4. **Backward snowballing**
   We make use of the references in all the papers selected by the active learning screening method. We then select the references within the paper that are useful for the systematic literature review with a stopping criterium (Van Haastrecht et al. [45]).
5. **Quality Assessment**
   We find out the quality/usefulness of each source and rank them based on a specific scoring metric.
6. **Data Extraction and Synthesis**
   We manually analyse the found literature and extract the meaningful data from those papers.
7. **Validation**
   For validation we find out the handling of bias in the data extraction and synthesis.

Throughout the project, we will deliver the following items: A systematic literature review protocol, database search query, an initial list, intermediate list, final list of papers included in the review, a spreadsheet with the results of the literature analysis and finally the thesis with the conclusions from the analysis and the answers to the research questions.

3.1 Protocol

In this section, we take the first step of the SYMBALS method, which is to define our research protocol.

3.1.1 Background rationale

The background is explained earlier in Chapter 2. To summarise, we are using SYMBALS as a systematic literature review method because it is novel and in a specific case been found to outperform another "state of the art model" [56]. We use this methodology for GenAI in code review because that has been an relatively low studied field.

The objective of this research is to find out if the SYMBALS method is useful in this case of systematic literature review. And to find out what the relatively novel field of GenAI means for the code review process.

3.1.2 Research questions

In Section 1.2, we formulated a main research question for our systematic literature review:

RQ1 What are the current research trends in applying GenAI to code reviews?

Although answering the main research question is the main goal of this systematic literature review, to give a more detailed view of what is important in this research question, we have made some more specific sub-research questions. These sub-research questions are as follows:

RQ1.1 For which code review tasks is AI assistance applied?

RQ1.2 Which AI techniques are used to support code review tasks?

RQ1.3 To what extent are AI-assisted code review support tools validated with users and ready for deployment?

The first sub-research question helps us with finding out exactly for which code review tasks GenAI techniques are used. The second sub-research question helps us with finding out what kind of AI models are used for code review related tasks. The third sub-research question helps us with finding out how ready for use, AI support tools are for code reviews. And how much these support tools are confirmed by the userbase.

3.1.3 Provisional query

This following section is not part of the SYMBALS method but was found to be useful. Before we set up the final query, we set up a provisional query to find out what kind of terms we had to use in the final query. This following provisional query was done in the ACM digital library and IEEE explore database:

```
document type (English) AND not retracted AND (code review)
AND (GenAI OR Generative OR "LLM")
```

We looked through a few papers full-text to determine the final terms for the provisional query. This was useful as well to deepen our knowledge in this field. We have used "LLM" in the provisional query because the most well-known AI model, ChatGPT, is an LLM model and we presumed that that would have big importance in this field as well.

3.1.4 Search strategy

The search strategy which we have used consists of using ACM digital library and IEEE explore. We have used these 2 databases because they were found to be the most relevant databases and the most compatible with the ASReview software. ACM digital library is a database focused on computing literature and IEEE xplore is a database partly focussed on information technology, two literature themes which have overlap with the subject of code review. A more general database we have tested and would have originally used was Google Scholar. The problems we encountered with this database are discussed in chapter 5.2.

3.1.5 Selection criteria

The selection criteria are written as a list of requirements where each criterium either includes or excludes a specific paper during the screening and backward snowballing phase.

Inclusion criteria

- I1: The research concerns research trends in applying GenAI in code reviews.

Exclusion criteria

Some of our exclusion criteria were inspired by the exclusion criteria of other systematic reviews using SYMBALS [45]:

- E1: The research does not concern anything related to code reviews.
- E2: The paper has been retracted.
- E3: another paper is more relevant to the specific research in the paper(if the author made a more updated version).
- E4: The research was automatically excluded due to its assessed irrelevance by the ASReview tool.
- E5: The research does not satisfy the database query criteria on English language.
- E6: The paper is of insufficient quality.
- E7: The paper does not contain at least one code review task where AI assistance is applied.
- E8: Research does not mention either an AI technique specifically to support code review tasks.
- E9: No duplicates.

Criterion E5 and E4 are directly taken from [45].

3.1.6 Selection procedure

The selection of papers was done during these 3 steps:

1. Active learning Screening
2. Backward Snowballing
3. Manually exclude during data extraction

During all these 3 steps the selection criteria were taken into account.

3.1.7 Data extraction, management, and synthesis strategy

After making the selection of papers, we perform data extraction. During data extraction, quality assessment is applied to further exclude papers based on inclusion and exclusion criteria.

For data extraction, we performed a full text review of all the selected papers. While reviewing we attempted to answer each of the 3 sub research questions for each selected paper. We wrote down the answers in a spreadsheet.

After a full review of all papers, we turned to the resulting spreadsheet to synthesize results. To get a clearer picture, we first normalised some of the answers in the spreadsheet. This was for example the case for the first research question, because of the multitude of different notations of tasks which are closely related. So we had to normalise these tasks description.

3.2 Database search

In our database search with the use of this search query in IEEE explore and ACM digital library, we came up with a total of 541 papers. We thought this was quite a lot of papers because, our rough expectation was that only around 100 papers would come up because the topic is very specific and novel.

The final query we had used in these 2 databases was:

```
((("Document Title":Code review OR
"Abstract":Code review OR "Document Title":Code inspection OR
"Abstract":Code inspection OR "Document Title":Code refactor* OR
"Abstract":Code refactor*)
```

```
AND ("Document Title":genai OR "Abstract":genai OR
"Document Title":generative ai OR "Abstract":generative ai OR
"Document Title":copilot OR "Abstract":copilot OR
"Document Title":LLM OR"Abstract":LLM OR
"Document Title":large language model OR "Abstract":large language model OR
"Document Title":artificial intelligence OR"Abstract":artificial intelligence)))
```

We developed this query on the basis of our provisional query. This query is different from the provisional query as it is a working query on its own instead of the more sketched provisional query. With the full text review of some of the papers in the provisional query we have added technical terms such as copilot, code inspection and code refactor to broaden our database search. We have written out LLM as Large language model to include papers which might mention a large language model once in their paper and never had to use the abbreviation, this was the case for AI as well.

3.3 Tool used for active learning

ASReview[1] is a novel tool which makes it possible to do the screening step in a systematic review quicker. Examples of use cases are in journalism to counter bias for specific subjects among sources or in the medical field to know if a specific medicine or treatment are working. It promises an up to 95% timesaving compared to screening every paper manually. It does this by making use of active learning. For every paper that a human points as relevant or irrelevant, the ASReview tool learns more about which paper it should show for review to the user. It does this in the ideal scenario until every paper that should have been found as relevant in a manual systematic review has been found relevant with the ASReview tool as well and with less manual labour.

3.3.1 Papers selected for training

Before running the ASReview tool, the tool has to take 10 papers to train on. Five of those papers had to be found relevant for our research and another five were not to be found relevant for our research. The five relevant papers were:

- Fine-tuning Large Language Models to Improve Accuracy and Comprehensibility of Automated Code Review [55]
- On the Use of ChatGPT for Code Review: Do Developers Like Reviews By ChatGPT? [51]
- How to refactor this code? An exploratory study on developer-ChatGPT refactoring conversations [3]
- LLaMA-Reviewer: Advancing Code Review Automation with Large Language Models through Parameter-Efficient Fine-Tuning [26]
- Automating Patch Set Generation from Code Reviews Using Large Language Models [33]

The five irrelevant papers:

- Refactoring Programs Using Large Language Models with Few-Shot Examples [36]
- Code Compliance Assessment as a Learning Problem [34]
- Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap [53]
- CoRA: decomposing and describing tangled code changes for reviewer [48]
- Combining LLM-Generated and Test-Based Feedback in a MOOC for Programming [13]

[1] https://asreview.readthedocs.io/en/stable/lab/about.html

Chapter 4

Results

In this chapter, we present the results of our systematic literature review. We first provide information on the number of papers selected through active learning, snowballing, and quality assessment. Then we dive into the findings of our analysis of the selected papers.

Figure 4.1, shows the number of papers we found at each step of our paper selection process. The final list of selected papers can be found in Table 4.1.

4.1 Active learning Screening

We took a conservative approach to labeling papers as 'irrelevant' in the active learning phase. Because in the ASReview tool itself, we can only see the abstract and the title it is hard to find out what the exact substance is in the paper. For example it may mention generative AI in a code review context briefly in the abstract. Does that mean a whole chapter talks about it, only a paragraph or is it just a suggestion for future research? To be sure in all three cases we have added the papers to the 'relevant' pile. The reason is that the expected number of papers to come out of the active learning phase is quite low. It is thus possible to manually weed out the papers that are not relevant in the data extraction phase.

We started the screening with active learning on a total of 541 papers that were found through our database search. The active learning phase was initiated by supplying a selection of 5 relevant and 5 irrelevant papers, as explained in Section 3. Throughout the active learning phase, we labeled another 143 papers as relevant or irrelevant. This brought down our initial list of 541 papers to a final list of 47 relevant papers. We found that nine of these were duplicates. There were thus 38 unique papers found to be relevant at the end of the active learning phase.

4.2 Backward Snowballing

During the backward snowballing phase, we looked at the reference list of each paper included at the end of the screening phase. Because ASReview ranks the papers on 'importance' it was recommended

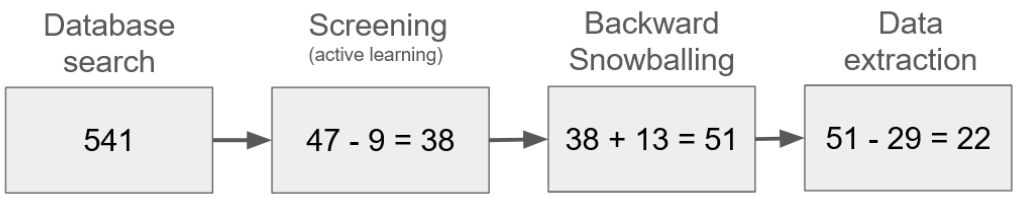

Figure 4.1: Number of paper through the systematic review

Table 4.1: Selected papers.

Source	Title	Year	Citations
[55]	Fine-tuning Large Language Models to Improve Accuracy and Comprehensibility of Automated Code Review	2024	18
[51]	On the Use of ChatGPT for Code Review: Do Developers Like Reviews By ChatGPT?		10
[3]	How to refactor this code? An exploratory study on developer-ChatGPT refactoring conversations	2024	31
[26]	LLaMA-Reviewer: Advancing Code Review Automation with Large Language Models through Parameter-Efficient Fine-Tuning	2023	116
[2]	Cultivating Software Quality Improvement in the Classroom: An Experience with ChatGPT	2024	1
[4]	Improving the Learning of Code Review Successive Tasks with Cross-Task Knowledge Distillation	2024	15
[22]	AUGER: automatically generating review comments with pre-training models	2022	64
[41]	Using Pre-Trained Models to Boost Code Review Automation	2022	188
[11]	Resolving Code Review Comments with Machine Learning	2024	24
[28]	An Empirical Study on Code Review Activity Prediction and Its Impact in Practice	2024	4
[42]	Code Review Automation: Strengths and Weaknesses of the State of the Art	2024	34
[24]	Towards Automated Code Reviews: Does Learning Code Structure Help?	2023	12
[57]	CoditT5: Pretraining for Source Code and Natural Language Editing	2022	107
[18]	Software Vulnerability and Functionality Assessment using LLMs	2024	19
[8]	Generative Pre-Trained Transformer (GPT) Models as a Code Review Feedback Tool in Computer Science Programs	2023	9
[54]	Generative AI for Pull Request Descriptions: Adoption, Impact, and Developer Interventions	2024	8
[25]	Automatic generation of pull request descriptions	2019	130
[49]	Divide-and-Conquer: Automating Code Revisions via Localization-and-Revision	2025[1]	0
[27]	Multi-Role Consensus Through LLMs Discussions for Vulnerability Detection	2024	12
[47]	AI-Assisted Assessment of Coding Practices in Modern Code Review	2024	12
[16]	Exploring the potential of ChatGPT in automated code refinement: An empirical study	2024	117
[38]	An analysis of the automatic bug fixing performance of ChatGPT	2023	442
[12]	VulRepair: A T5-Based Automated Software Vulnerability Repair	2022	236

[1] This paper appeared online in 2024, but was officially publised in 2025.

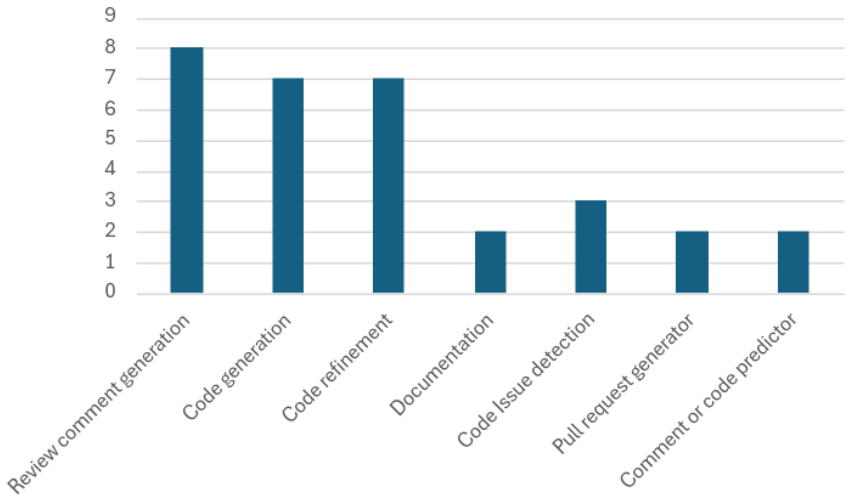

Figure 4.2: Number of papers in which each normalised task is discussed.

to do the backwards snowballing phase in the order of the list given at the end of the screening phase. We stopped after 10 consecutive papers where not a single reference was added to the review pile.

Because Generative AI in code review is a relatively new concept, papers with publication dates before 2014 were not included in this phase. We consider 2014 as a safe cut-off date, since the breakthrough to the general public of LLMs for coding tasks only happened in 2021, with the publication of Chen et al. [5] on the Codex model and the launch of the Github Copilot programming assistant [14].

In the backward snowballing step we looked at the reference list of all the 38 papers found to be useful during the active learning phase. Just as in the screening phase we only looked at the title and abstract. The total number of references in the 38 papers was 1531. In this phase we only found 13 additional papers to be useful and adhering to the requirements we have set up earlier, bringing us to a total of 51 papers after snowballing.

4.3 Data extraction

During the data extraction, 29 papers were found to be not useful enough. The reason that these papers were not discarded earlier is that during the active learning screening, we could only see the abstract and title and we could not see what the whole paper was about. While doing the data extraction we found out that some papers were not papers at all and were just a small part in a book, or a only a proposal. Thus these papers were of insufficient quality for the systematic review. For some paper the full text version was not able to be found at all. With the removal of these papers from the review pile, we ended up with 22 papers in total at the end of the data extraction.

4.4 General findings

In this section, we provide answers to our research questions on the basis of our analysis results.

In addition, we report about the distribution of the papers over publication years, number of authors, type of paper and number of citations.

4.4.1 RQ1: For which code review tasks is AI assistance applied?

Figure 4.2 shows the number of papers in which each task is mentioned.

The code review tasks were described within the papers by a diversity of terms. To produce a comprehensive chart, we normalises the terms given for the code review tasks. For instance, the terms ‘comment predictor’ and ‘code predictor’ were normalised to ‘comment or code predictor’.

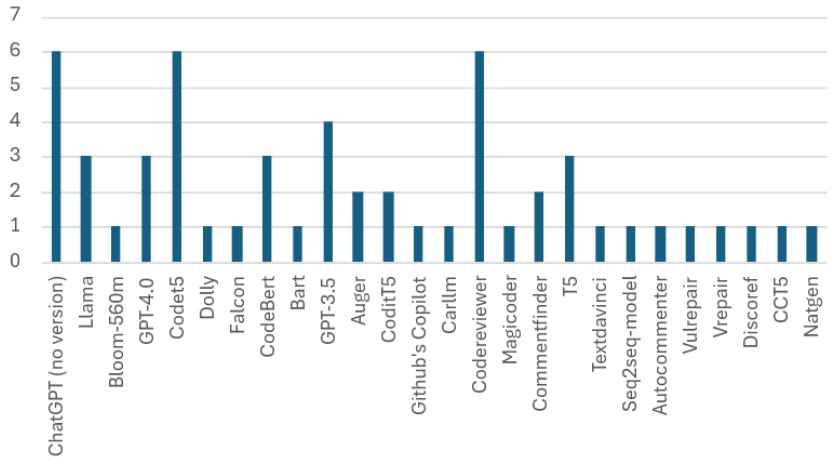

Figure 4.3: Number of papers in which each AI technique is mentioned.

Note that the number of times all the different code review tasks are used exceeds the number of papers taken into account in this research. The reason is that in some papers multiple code review tasks have been mentioned within one paper.

The task which is most mentioned in the selected papers with eight occurrences is 'review comment generation'. The second most mentioned tasks with seven occurrences are 'code generation' and 'code refinement'. The 'code issue detection' task was mentioned in three papers. The following tasks are each mentioned in two papers: 'pull request generator', 'comment or code predictor' and 'documentation'.

4.4.2 RQ2: Which AI techniques are used to support code review tasks?

Figure 4.3 shows the number of papers in which each AI technique is mentioned.

ChatGPT The most mentioned AI technique, with ten unique occurrences in total, is ChatGPT, the LLM made by OpenAI. Not all papers provide clarity on which specific version of ChatGPT is used. Three papers explicitly refer to version 4.0, while version 3.5 is mentioned in four papers. In six papers there was no version of ChatGPT mentioned. According to OpenAI themselves the following is said: "ChatGPT is designed to understand and respond to user questions and instructions by learning patterns from large amounts of information, including text, images, audio, and video."[29] This means that although ChatGPT is the most often used AI technique for code review in the selection, the model itself was not even designed for this purpose and more as a general purpose model.

CodeT5 The second most frequently mentioned AI technique is CodeT5, with six occurrences. CodeT5 is an AI technique which has been produced on the basis of the T5 structure (Text-to-Text Transfer Transformer). CodeT5 was developed by Salesforce Research.[1] CodeT5 has not been trained specifically for the purpose of code review but it has been designed purely for code related tasks [50].

Codereviewer Codereviewer is mentioned in six papers. as said by the proposers of this technique, "CodeReviewer, a pre-trained model that utilizes four pre-training tasks tailored specifically for the code review senario" Li et al. [23]. Thus this AI technique is designed for code review related tasks.

LLama The AI technique Llama only occurred in three papers. LLama is an LLM model made by the MetaAI company in 2023.[2] This is a paid service[3] by that same company. Two out of the three papers that make use of this model, make use of LLama-reviewer which is trained for code review tasks as mentioned by Lu et al. [26]: "LLamA-Reviewer is specifically designed to automate three core tasks integral to the code review process, namely review necessity prediction, code review comment generation, and code refinement". The Llama technique used in one paper which was not Llama-reviewer is a general purpose model.

[1] https://github.com/salesforce/CodeT5?tab=readme-ov-file

[2] https://www.cnbc.com/2023/02/24/mark-zuckerberg-announces-meta-llama-large-language-model.html

[3] https://www.llama.com/llama-downloads/?utm_source=llama-home-hero&utm_medium=llama-referral&utm_campaign=llama-utm&utm_offering=llama-downloads&utm_product=llama

Apart from these frequently mentioned techniques, the following techniques are also mentioned.

CodeBert In our selection CodeBert is mentioned three times. This is a model focused on code related tasks [10].

Auger Auger is mentioned in two papers. Auger is based on the T5 architecture. As its name says (AUtomatically GEnerating Review comments) [22], it is a technique designed for code review related tasks.

Commentfinder Commentfinder is mentioned in two papers. It is said by Hong et al. [17]"CommentFinder is a retrieval-based approach to recommend code review comment". Thus it is the case that this is a technique focused on code review related tasks.

T5 Three papers make use of T5. T5 is quite an old technique originating out of 2019. It is a general purpose technique [31]. In our research it is the case that this architecture is used to make 2 code review related tasks techniques in two different papers.

CoditT5 Two papers make use of the CoditT5. As said by Zhang et al. [57]. "CoditT5, a large language model for software-related editing tasks that is pretrained on large amounts of source code and natural language comments." We can presumably conclude from this that it is a technique made on the basis of handling code related tasks.

The least occurring AI techniques, with each a single occurrence in the selection of papers are: Bloom-560m, Dolly, Falcon, Bart, GitHub's Copilot, CAR-LLM, Magicoder, textdavinci, an sequence to sequence model, code reviser, AutoCommenter, Vulrepair, Vrepair, Discoref, CCT5 and natgen.

Bloom-560m Bloom-560m is an AI technique based on the more general Bloom LLM. It is a general purpose model.[4]

Dolly Dolly's AI model is according to the makers "the first LLM trained on an open-source,human generated instruction dataset" Conover et al. [7]. It says that it exhibits "ChatGPT-like human interactivity". This technique has been designed with a general purpose target in mind.

Falcon Falcon is an AI model that only occurred in one paper. This LLM has been trained on the AWS cloud (parent company Amazon). This model is a more general purpose technique. The makers of this model say themselves its "one of the three best language models in the world along with GPT-4 and PaLM-2-Large" Almazrouei et al. [1].

Github's copilot Github's copilot is an AI technique which has been designed for the purpose of helping with code related tasks.[5]

CAR-LLM CAR-LLM is according to the makers of this technology: "a novel fine-tuned LLM that has the ability to improve not only the accuracy but, more importantly, the comprehensibility of automated code review" Yu et al. [55] and thus a technique for code reviews.

Magicoder Magicoder is a code focused AI technique. This is shown on its own Github page: "enlightening LLMs with open-source code snippets for generating low-bias and high-quality instruction data for code" Wei et al. [52]. It is thus a technique focused on code related tasks.

Text-Davinci Text-Davinci is a general purpose technique. The technique used in the paper in our selection was "Text-davinci-003" which on the moment of writing been deprecated by OpenAI.

Sequence to sequence model In one paper a sequence to sequence model was specifically made in the paper itself. The goal of this model is to "generate descriptions for pull requests from their commit messages" Liu et al. [25]. It is thus a technique specifically made for code reviews.

AutoCommenter AutoCommenter is a model which is used by Google's developers. It is a technique which enforces "best practises" of coding [47]. It is thus a technique focusing on code review.

Vulrepair Vulrepair is a technique based on the T5 architecture. It is a technique focused on fixing security bugs [12]. This can been seen as a technique focused on a code review task.

Vrepair Vrepair is a technique focused on fixing security bugs [6]. This can been seen as a technique focused on a code review task.

[4] `BigScience,BigScienceLanguageOpen-scienceOpen-accessMultilingual(BLOOM)LanguageModel.International, May2021-May2022`

[5] `https://github.com/features/copilot`

Discoref Discoref is a technology which as said in the paper itself: "our approach, we utilize a cascade of models to enhance both comment generation and code refinement models" [4]. Thus it is a code review technique.

CCT5 CCT5 is a technique which in its training step uses a dataset of "code commits and commit message" Wang et al. [49]. Thus it is a technique focused on code review.

Natgen Natgen, is a technique discussed in [49]. It is a technique that could be able to write human like code paraphrasing from [49]. It is an technique focusing in code related tasks.

The following results are shown in Table 4.2 as well. In total there were 22 papers researched. There were 12 papers which only have used 1 AI technique. 2 Papers mention 2 AI techniques. 3 papers mention 3 different AI techniques. One paper mentions 5 AI techniques. One paper mentions 7 AI techniques. One paper mentions 10 AI techniques. Finally one paper mentions 14 different AI techniques.

Table 4.2: Number of papers mentioning certain numbers of AI techniques.

Number of AI techniques	1	2	3	4	5	7	10	14
Number of papers	12	2	3	1	1	1	1	1

Table 4.3 shows the distribution between techniques which are a part of general purpose techniques, code related techniques or finally techniques focussed on code review.

Table 4.3: Number of papers where models are mentioned of various levels of specificity.

General purpose	Code related	Code review
19 papers	14 papers	21 papers

4.4.3 RQ3: To what extent are AI-assisted code review support tools validated with users and ready for deployment?

Our third research sub-question (SRQ3) is:

> To what extent are AI-assisted code review support tools validated with users and ready for deployment?

We provide an qualitative answer to this question.

There were only three papers where the use of an AI technique was found to be ready for use and to some extent accepted by the people who ultimately have to use them. One of them was a technique being used by Google which ended up being used in $7,5\%$ of their code-review comments [11].

The second one was for a very specific coding course validated by a group of 74 students [8]. Of this group of 74 students, 66 of them (89.2%) found that "the 'reviewer' had the proper knowledge to review their code" [8] by saying 'yes'. 78.3% of the 74 students said as well that they either agreed or strongly agreed to the question "Would you want this reviewer to review your code again in the future?" [8].

The final paper where the new support tool was validated by the users and ready for deployment is "AI-Assisted Assessment of Coding Practices in Modern Code Review" [47]. In this paper an AI technique called 'AutoCommenter' was deployed in Google slowly to increasing numbers of developers. The authors of this papers only increased the size of this group to half of all the developers after reaching the 80% useful ratio target. Before they reached this target they made their 'AutoCommenter' better at least partially to keep the "developers trust" [47]. The developer feedback was tracked every month in positive to negative ratio. At the last month only 5% of the total feedback for that month was negative [47].

None of the other selected papers reported on validation experiments with users and acceptance into a production environment.

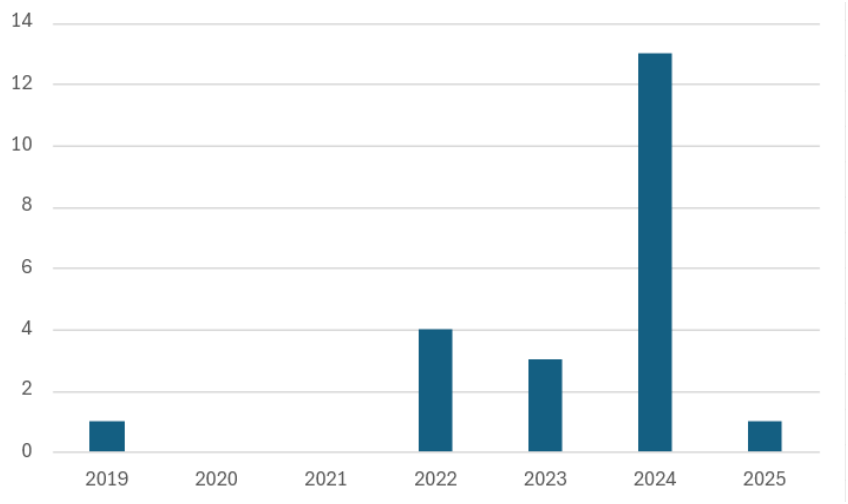

Figure 4.4: Publishing year of each paper.

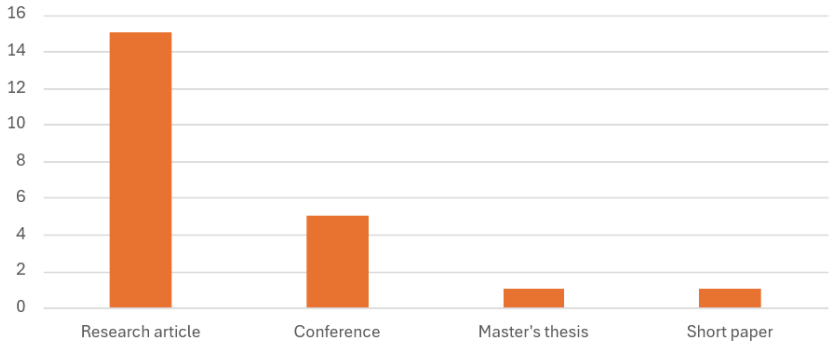

Figure 4.5: The types of papers.

4.4.4 Distribution of papers

In this part we discuss the analysis of the data perse not important for the sub-research questions. But is important for the later analysis if the methodology we followed was useful for the results we achieved.

Year of publication

Figure 4.4 shows the number of papers which were published each year. Most papers (13) in our selection were published in 2024. Three of the papers were published in 2023 and another four paper were published in 2022. Only one paper was published in 2019 and only one paper was from 2025.

Type of paper

Figure 4.5 shows the type of papers in which each AI technique is mentioned. We found that 15 papers have the form of a research article. The second most occurring type of paper is a conference article with six. There is one paper written in the format of a Master's thesis [27] and one in the format of a short paper [51].

Number of authors per paper

Figure 4.6 shows the number of authors for each paper. Eleven of the papers researched have the authorship of a small group of around five to ten people. Nine papers were authored by a group of less than five authors. The least amount of papers were written by a group consisting of 11 or more people. This occurred only in two papers.

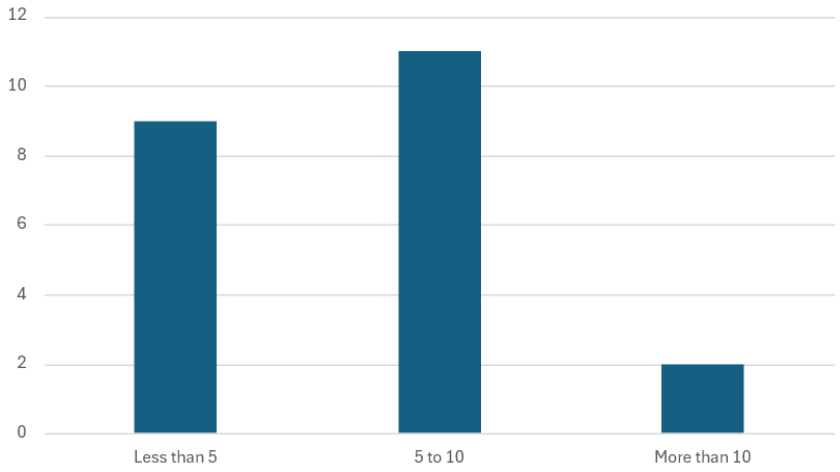

Figure 4.6: Number of authors in each paper.

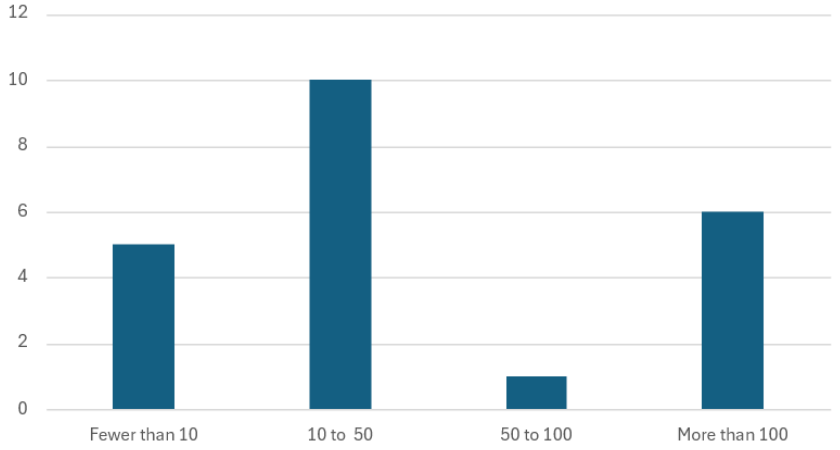

Figure 4.7: Number of citations in google scholar for each paper.

Number of citations

Figure 4.7 shows the number of citations other papers have made for each paper in our selection, according to Google Scholar. To begin with, ten papers have only been cited by other papers fewer than ten times. Seven papers have been cited between 10 to 50 times, Only one paper has been cited 50 to 100 times. Finally, Six papers have been cited more than 100 times. The papers which have been cited the most are: [26],[41],[57],[25],[16] and [12].

4.5 Validation

Could our selection have missed some papers? The search term we have used was very broad. we were treating the rejection of papers at the conservative side. We have even used Backward snowballing to possibly find out papers we might have missed.

Could our opinion have taken a role in determining the answer to the research question? we could have influenced the results of answering this research question. because we have determined 3 sub-research question before starting with the systematic review, we think we have excluded possible bias during the research for answering the research question.

Chapter 5

Discussion

In this chapter, we try to interpret our results and put them into context. We first do this from the perspective of GenAI-assisted code review. Secondly, we discuss the implications for AI-assisted structured literature reviews. Finally, we discuss study limitations.

5.1 GenAI-assisted code review

We found that ChatGPT is the most often used GenAI model used to assist in code review. This is a general purpose model. A reason for why this model is used so often could be that it is just the model most people have knowledge of. Because generative AI in most researcher's mind could be a new concept the usage of an AI model which most people have at least an elementary amount of knowledge of could be the explanation.

Even though ChatGPT is the most often used technology for GenAI in code review, the number of technologies used which was designed with code review in mind is higher than the technologies which are only code related or general purpose. So this could possibly point at an increasing amount of specialization of AI models in this field. The results of the specificity of each technique can be seen in Table 4.3.

Another interesting point is that big tech companies part of the FAANG group namely Facebook and Amazon, have their own AI technologies which have been found to be useful enough to test on code review related tasks. This might point to a competition between these companies to make the best versatile LLMs.

The various companies that have viable LLM's also might point to that this technique is in its beginning phase and that in the future there might be fewer LLM's in use. This could be comparable to when a lot of companies started making search engines in the early 2000s and late 1990s. Which today has largely converged to Google dominating the search engine market with 79.5% of desktop users making use of Google as their search engine [15].

An interesting results for the first sub-research question is that in a large number of tasks, researchers are already trying to fully automate large parts of the code review process. For example automatically generating review comments or generating wholly new code.

In the results we can also see that the ready for deployment and acceptance by users of AI models in code review is quite low. Only 3 out of 22 papers have validated their techniques with the end users. This could point at that AI models in code review is in its phase that models able to be used in a "normal" setting are still in development.

If we compare this with a model quite often used in novel techniques in ICT called Gartner hype cycle for emerging technologies,[1] we can presume that this technique in this usage is either at its initial stage "technology trigger" or at its "peak of inflated expectations". The reasoning behind this is that only a small percentage of investigated papers mention that the technique is used in real life usecases and

[1] https://www.gartner.com/en/newsroom/press-releases/2024-08-21-gartner-2024-hype-cycle-for-emerging-technologies-highlights-developer-productivity-total-experience-ai-and-security

presumably was used after the publishing of the paper. A majority of the other papers test models based on performance without validating it with the end-users if it useful.

We now use the technology radar of Thoughtworks. which is a "snapshot of tools, techniques, platforms, languages and frameworks. This knowledge-sharing tool is based on our global teams' experience and highlights things you may want to explore on your projects" [21]. It is thus a report that comes out every few months to look at a lot of new technology and tries to answer the question if these technologies should be adopted, trialed, assessed or be on hold in a business. If we compare our subject with a similar subject called "Replacing pair programming with AI"[2], we see that this technology is part of a group of technologies which are the least far in development to be used within a project in a business. This technology is in the hold column.

5.2 AI-assisted Structured Literature Reviews

We have used the novel method of SYMBALS as much as possible according to the inventor of this methodology. We found out that database selection might be an issue for specific fields because of incompatibility with the software used by van Haastrecht et al..

As has been mentioned in the related work in Chapter 2, the ASReview tool did come up short in some aspects. The first shortcoming was the difficulty of exporting the search in the database into the correct csv file. In Sijtsma et al. [37] they warned for this and said that the database search has to be extensively checked if it was exported correctly.

When selecting the 5 papers to train the ASReview tool on we added a paper which was later excluded from the final pile. The reason for this exclusion was that the paper was of too low quality to be added. This mistake can be attributed to our lack of experience with the methodology. Lack of experience with the methodology was also identified as a risk by Sijtsma et al. [37].

While we expected that most papers would have been research articles or conferences, we found it interesting that a Master's thesis turned out to be included in the final review pile.

The number of papers with a group of 5 or more people as author with 15 papers was higher than the number of papers having 4 or less authors with 9 papers. In our opinion, this could possibly point at ASReview putting more importance on papers with larger groups of authors. Alternatively, papers with long author lists may be overrepresented in this research topic because of the effort involved in developing tooling and running experiments with GenAI for code reviews.

5.3 Limitations

By using the ASReview model described in the SYMBALS not all databases are compatible. For instance, the Google Scholar database does not have an extraction method of receiving all of the abstracts and titles of a database search, while these are essential for the ASReview model to work.

The ability to fully read all of the papers found by the initial database search is also not guaranteed. The backwards snowballing step of SYMBALS was found to be quite time consuming as well. The references in papers which eventually were labelled to be relevant were hard to get working with the ASReview model. In our opinion if someone wants to do a systematic review who does not have at least medium knowledge about the subject, should not start doing such a systematic review.

The usage of ASReview in general was more difficult than expected which makes true what was mentioned by Sijtsma et al. [37] "Use of the program requires skill from the researcher to drive the algorithm and expert knowledge to start the process and expert knowledge to interpret the results."

At the end of the screening phase, 9 out of the 47 papers were duplicates. What we later found out is that we should have looked manually at the initial database search for duplicates and removed them as said bySijtsma et al. [37]. We think however that this may counter the in our opinion positives of using this SYMBALS method which is spending a smaller amount of time on the screening phase in a systematic review.

[2] `https://www.thoughtworks.com/radar/techniques/summary/replacing-pair-programming-with-ai`

Chapter 6

Conclusion

We conclude this research with a short formulation of the answer to our main research question. We also briefly indicate wider implications of our research and suggest some avenues of future work.

6.1 Answer main research question

The overall research question as formulated in our Introduction chapter was:

RQ1 What are the current research trends in applying GenAI to code reviews?

Firstly, we can conclude that the field of generative AI in code review is a rising field that is still in its infancy. This can be seen from the fact that the total number of papers we identified is fairly limited (22), that all of these papers are quite recent (most from 2024), and that only about 10% of these papers present a technique that is validated by its users and seen as ready for deployment.

Secondly, we have observed that the tasks most commonly automated with GenAI for code review are review comment generation, code generation, and code refinement.

Thirdly, the GenAI techniques being experimented with covers a very wide range, from general-purpose models to code-related models, and even code-review specific models. The last category being the largest may point to increasing specialisation in the field.

6.2 Implications for SLR's using SYMBALS

SYMBALS is a methodology that in our opinion makes the process of doing a systematic review faster to do, but does require intensive study into the methodology. Because at a lot of points ASReview could receive the wrong data or could output the wrong data, which the user needs to be able to filter out.

Still, we think that the SYMBALS methodology could be used as a systematic review method in other fields. It was already earlier proven that this methodology could be done outside of a computer science contextvan den Bulk et al. [43].

It has to be said that in our opinion using the methodology on systematic review that are smaller in scale would negate the benefits received by using the SYMBALS method. Reviews consisting of fewer than 100 papers at the stage of database search would not benefit from SYMBALS.

It also has to be said that this methodology does not work with every database, if the recommended screening model is used.

6.3 Future work

We think that the introduction of GenAI in code reviews has just begun and think that if a new systematic review will be done in a few years it will come up with more data. As generative AI models have just recently become popular in use and research. This can been seen in the form that OpenAI's ChatGPT has only been released for public use in 2022 together with Dolly.

Future work should also be more done on the validation of GenAI models in code review tasks instead of only comparing AI models with each other purely on performance.

We propose as well to validate this method via replication. This can be done by another person to check if the exercise of our methodology is validated.

Bibliography

[1] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

[2] Eman Abdullah AlOmar and Mohamed Wiem Mkaouer. Cultivating software quality improvement in the classroom: An experience with chatgpt. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 1–10. IEEE, 2024.

[3] Eman Abdullah AlOmar, Anushkrishna Venkatakrishnan, Mohamed Wiem Mkaouer, Christian Newman, and Ali Ouni. How to refactor this code? an exploratory study on developer-chatgpt refactoring conversations. In *Proceedings of the 21st International Conference on Mining Software Repositories*, pages 202–206, 2024.

[4] Oussama Ben Sghaier and Houari Sahraoui. Improving the learning of code review successive tasks with cross-task knowledge distillation. *Proceedings of the ACM on Software Engineering*, 1(FSE): 1086–1106, 2024.

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[6] Steve Kommrusch chenzimin and Martin Monperrus. Vrepair, 2024. URL `https://github.com/ASSERT-KTH/VRepair`. Last accessed 30 June 2025.

[7] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free Dolly: Introducing the world's first truly open instruction-tuned LLM, 2023. URL `https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm`. Last accessed 30 June 2025.

[8] Aaron S Crandall, Gina Sprint, and Bryan Fischer. Generative pre-trained transformer (gpt) models as a code review feedback tool in computer science programs. *Journal of Computing Sciences in Colleges*, 39(1):38–47, 2023.

[9] Patrick Samson Udama Eneche, Funda Atun, Yijian Zeng, and Karin Pfeffer. Robust drivers of urban land surface temperature dynamics across diverse landscape characters: An augmented systematic literature review. *Ecological Indicators*, 163:112056, 2024.

[10] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

[11] Alexander Frömmgen, Jacob Austin, Peter Choy, Nimesh Ghelani, Lera Kharatyan, Gabriela Surita, Elena Khrapko, Pascal Lamblin, Pierre-Antoine Manzagol, Marcus Revaj, et al. Resolving code review comments with machine learning. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, pages 204–215, 2024.

[12] Michael Fu, Chakkrit Tantithamthavorn, Trung Le, Van Nguyen, and Dinh Phung. Vulrepair: a t5-based automated software vulnerability repair. In *Proceedings of the 30th ACM joint european software engineering conference and symposium on the foundations of software engineering*, pages 935–947, 2022.

[13] Hagit Gabbay and Anat Cohen. Combining llm-generated and test-based feedback in a mooc for programming. In *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, pages 177–187, 2024.

[14] Dave Gershgorn. Github and openai launch a new ai tool that generates its own code, 2025. URL `https://www.theverge.com/2021/6/29/22555777/github-openai-ai-tool-autocomplete-code`. Last accessed 30 June 2025.

[15] Statcounter Globalstats. Search engine market share worldwide, 2025. URL `https://gs.statcounter.com/search-engine-market-share/desktop/worldwide`. Last accessed 1 July 2025.

[16] Qi Guo, Junming Cao, Xiaofei Xie, Shangqing Liu, Xiaohong Li, Bihuan Chen, and Xin Peng. Exploring the potential of chatgpt in automated code refinement: An empirical study. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13, 2024.

[17] Yang Hong, Chakkrit Tantithamthavorn, Patanamon Thongtanunam, and Aldeida Aleti. Commentfinder: a simpler, faster, more accurate code review comments recommendation. In *Proceedings of the 30th ACM joint European software engineering conference and symposium on the foundations of software engineering*, pages 507–519, 2022.

[18] Rasmus Ingemann Tuffveson Jensen, Vali Tawosi, and Salwa Alamir. Software vulnerability and functionality assessment using llms. In *2024 IEEE/ACM International Workshop on Natural Language-Based Software Engineering (NLBSE)*, pages 25–28. IEEE, 2024.

[19] Samar Samir Khalil, Noha S Tawfik, and Marco Spruit. Exploring the potential of federated learning in mental health research: a systematic literature review. *Applied Intelligence*, 54(2):1619–1636, 2024.

[20] Barbara Kitchenham. Guidelines for performing systematic literature reviews in software engineering. *in EBSE Technical Report EBSE-2007-01*, page 7, 01 2007.

[21] Rachel Laycock, Martin Fowler, Bharani Subramaniam, Birgitta Böckeler, Bryan Oliver, Camilla Falconi Crispim, Chris Chakrit Riddhagni, Effy Elden, Erik Doernenburg, James Lewis, Ken Mugrage, Maya Ormaza, Mike Mason, Neal Ford, Ni Wang, Nimisha Asthagiri, Pawan Shah, Selvakumar Natesan, Shangqi Liu, Vanya Seth, and Will Amaral. Technology radar an opinionated guide to today's technology landscape, 2025. URL `https://www.thoughtworks.com/radar`. Last accessed 30 June 2025.

[22] Lingwei Li, Li Yang, Huaxi Jiang, Jun Yan, Tiejian Luo, Zihan Hua, Geng Liang, and Chun Zuo. Auger: automatically generating review comments with pre-training models. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1009–1021, 2022.

[23] Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, et al. Codereviewer: Pre-training for automating code review activities. *arXiv preprint arXiv:2203.09095*, 2022.

[24] Hong Yi Lin and Patanamon Thongtanunam. Towards automated code reviews: Does learning code structure help? In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 703–707. IEEE, 2023.

[25] Zhongxin Liu, Xin Xia, Christoph Treude, David Lo, and Shanping Li. Automatic generation of pull request descriptions. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 176–188. IEEE, 2019.

[26] Junyi Lu, Lei Yu, Xiaojia Li, Li Yang, and Chun Zuo. Llama-reviewer: Advancing code review automation with large language models through parameter-efficient fine-tuning. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pages 647–658. IEEE, 2023.

[27] Zhenyu Mao. *Multi-role consensus through llms discussions for vulnerability detection*. CSCE, Waseda University, 2024.

[28] Doriane Olewicki, Sarra Habchi, and Bram Adams. An empirical study on code review activity prediction and its impact in practice. *Proceedings of the ACM on Software Engineering*, 1(FSE): 2238–2260, 2024.

[29] OpenAI. How chatgpt and our foundation models are developed, 2025. URL `https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-foundation-models-are-developed`. Last accessed 1 July 2025.

[30] Matthew J Page, Joanne E McKenzie, Patrick M Bossuyt, Isabelle Boutron, Tammy C Hoffmann, Cynthia D Mulrow, Larissa Shamseer, Jennifer M Tetzlaff, Elie A Akl, Sue E Brennan, et al. The prisma 2020 statement: an updated guideline for reporting systematic reviews. *bmj*, 372, 2021.

[31] Paperswithcode. T5, 2025. URL `https://paperswithcode.com/method/t5`. Last accessed 30 June 2025.

[32] Yazhuo Quan, Tetiana Tytko, and Bronson Hui. Utilizing asreview in screening primary studies for meta-research in sla: A step-by-step tutorial. *Research Methods in Applied Linguistics*, 3(1):100101, 2024.

[33] Md Tajmilur Rahman, Rahul Singh, and Mir Yousuf Sultan. Automating patch set generation from code reviews using large language models. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 273–274, 2024.

[34] Neela Sawant and Srinivasan H Sengamedu. Code compliance assessment as a learning problem. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 445–454. IEEE, 2023.

[35] Christoph Schröer, Felix Kruse, and Jorge Marx Gómez. A systematic literature review on applying crisp-dm process model. *in Procedia Computer Science*, 181:528, 2021.

[36] Atsushi Shirafuji, Yusuke Oda, Jun Suzuki, Makoto Morishita, and Yutaka Watanobe. Refactoring programs using large language models with few-shot examples. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, pages 151–160. IEEE, 2023.

[37] Frans Sijtsma, Liselotte Vreeling, Per Angelstam, Hacen El-Hacen, Martijn van der Heide, Ina Horlings, Theunis Piersma, and Pablo Tittonell. *Systematic and fast scientific literature review for policy purposes: An exploration of using ASReview software about the effectiveness of policy instruments promoting sustainable agriculture*. University of Groningen, 2021.

[38] Dominik Sobania, Martin Briesch, Carol Hanna, and Justyna Petke. An analysis of the automatic bug fixing performance of chatgpt. In *2023 IEEE/ACM International Workshop on Automated Program Repair (APR)*, pages 23–30. IEEE, 2023.

[39] Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Raula Gaikovina Kula, Norihiro Yoshida, Hajimu Iida, and Ken-ichi Matsumoto. Who should review my code? a file location-based code-reviewer recommendation approach for modern code review. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, page 141. IEEE, 2015.

[40] Patanamon Thongtanunam, Chanathip Pornprasit, and Chakkrit Tantithamthavorn. Autotransform: Automated code transformation to support modern code review process. *in Proceedings of the 44th international conference on software engineering*, 02 2022.

[41] Rosalia Tufano, Simone Masiero, Antonio Mastropaolo, Luca Pascarella, Denys Poshyvanyk, and Gabriele Bavota. Using pre-trained models to boost code review automation. In *Proceedings of the 44th international conference on software engineering*, pages 2291–2302, 2022.

[42] Rosalia Tufano, Ozren Dabić, Antonio Mastropaolo, Matteo Ciniselli, and Gabriele Bavota. Code review automation: strengths and weaknesses of the state of the art. *IEEE Transactions on Software Engineering*, 50(2):338–353, 2024.

[43] Simone van den Bulk, Amy Manten, Tobias N Bonten, and Ralf E Harskamp. Chest pain in primary care: a systematic review of risk stratification tools to rule out acute coronary syndrome. *The Annals of Family Medicine*, 22(5):426–436, 2024.

[44] Max Van Haastrecht, Guy Golpur, Gilad Tzismadia, Rolan Kab, Cristian Priboi, Dumitru David, Adrian Răcătăian, Louis Baumgartner, Samuel Fricker, Jose Francisco Ruiz, et al. A shared cyber threat intelligence solution for smes. *Electronics*, 10(23):2913, 2021.

[45] Max van Haastrecht, Injy Sarhan, Bilge Yigit Ozkan, Matthieu Brinkhuis, and Marco Spruit. Symbals: A systematic review methodology blending active learning and snowballing. *in Frontiers in research metrics and analytics*, 6:1–13, 2021.

[46] Max Van Haastrecht, Bilge Yigit Ozkan, Matthieu Brinkhuis, and Marco Spruit. Respite for smes: A systematic review of socio-technical cybersecurity metrics. *Applied sciences*, 11(15):6909, 2021.

[47] Manushree Vijayvergiya, Małgorzata Salawa, Ivan Budiselić, Dan Zheng, Pascal Lamblin, Marko Ivanković, Juanjo Carin, Mateusz Lewko, Jovan Andonov, Goran Petrović, et al. AI-assisted assessment of coding practices in modern code review. In *Proceedings of the 1st ACM International Conference on AI-Powered Software*, pages 85–93, 2024.

[48] Min Wang, Zeqi Lin, Yanzhen Zou, and Bing Xie. Cora: Decomposing and describing tangled code changes for reviewer. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1050–1061. IEEE, 2019.

[49] Shangwen Wang, Bo Lin, Liqian Chen, and Xiaoguang Mao. Divide-and-conquer: Automating code revisions via localization-and-revision. *ACM Transactions on Software Engineering and Methodology*, 34(3):1–26, 2025.

[50] Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *EMNLP*, 2021.

[51] Miku Watanabe, Yutaro Kashiwa, Bin Lin, Toshiki Hirao, Ken'Ichi Yamaguchi, and Hajimu Iida. On the use of chatgpt for code review: Do developers like reviews by chatgpt? In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pages 375–380, 2024.

[52] Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. magicoder, 2024. URL `https://github.com/ise-uiuc/magicoder`. Last accessed 30 June 2025.

[53] Xingyu Wu, Sheng-hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Evolutionary computation in the era of large language model: Survey and roadmap. *IEEE Transactions on Evolutionary Computation*, 2024.

[54] Tao Xiao, Hideaki Hata, Christoph Treude, and Kenichi Matsumoto. Generative AI for pull request descriptions: Adoption, impact, and developer interventions. *Proceedings of the ACM on Software Engineering*, 1(FSE):1043–1065, 2024.

[55] Yongda Yu, Guoping Rong, Haifeng Shen, He Zhang, Dong Shao, Min Wang, Zhao Wei, Yong Xu, and Juhong Wang. Fine-tuning large language models to improve accuracy and comprehensibility of automated code review. *ACM transactions on software engineering and methodology*, 34(1):1–26, 2024.

[56] Zhe Yu and Tim Menzies. Fast2: An intelligent assistant for finding relevant papers. *in Expert Systems with Applications*, 120:57–71, 2019.

[57] Jiyang Zhang, Sheena Panthaplackel, Pengyu Nie, Junyi Jessy Li, and Milos Gligoric. Coditt5: Pretraining for source code and natural language editing. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022.