Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
June 2024, vol. 29, no. 1, pp. 68–77
https://doi.org/10.2478/acss-2024-0009
https://content.sciendo.com

Generative Artificial Intelligence Use in Optimising Software Engineering Process: A Systematic Literature Review

Uldis Karlovs-Karlovskis*
Riga Technical University, Riga, Latvia

***Abstract* – Generative AI is only a few years old but already being applied in Software Engineering (SE). This literature review examines the most popular SE sub-fields of such cases and research methods that are typically used. 117 studies starting from 2020 have been assessed, and literature review has shown that the most active research is ongoing in the code generation area. It is not clearly defined by researchers, but the majority of the methods can be assumed as experiments. It is concluded that researchers often do not define the used research method with exclusions such as literature review or opinion survey. However, different validation methods are highly valued and applied thoroughly.**

***Keywords* – Agile, DevOps, generative AI, software engineering, systematic literature review.**

I. Introduction

Generative Artificial Intelligence (Generative AI) is only a few years old but already being applied in Software Engineering (SE). With this rise of Generative AI in the post-COVID-19 era, a decline in the focus of researching Software Engineering (SE) by scientists has been observed. Table I shows data on how the amount of research in SE has been continuously growing since 2014 but then started declining in 2021 and continued to decline throughout 2022 and 2023. However, the same research but with a focus on Generative AI is growing. The goal of this paper is to understand how much of this research is dedicated to optimising the SE practices particularly. This literature review examines the most popular SE sub-fields of such cases and the research methods that are typically used.

TABLE I
Historical Data of Software Engineering Research Studies

Year/ source	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
Studies of Generative AI and SE in Scopus.com	2	2	3	8	10	16	24	20	17	45
Studies of SE in Scopus.com	8189	8222	9107	9256	8666	9018	9186	7569	5721	5496
Studies of SE in Google Scholar	16 800	20 200	23 200	25 000	33 300	43 000	53 700	64 500	57 000	44 400

This systematic literature review continues the work that G. Giray [1] started but focuses on Generative AI applications for SE specifically. Since the interest in optimising SE as such seems to be declining, it is important to know who the most active researchers are and what methods they use in their work. Therefore, the study researches SE sub-field popularity, active researchers and used research methods.

117 studies starting from 2020 have been assessed, and literature review has shown that the most active research is ongoing in the code generation area. The underrepresented SE sub-fields are found to be Build, Release and Deploy.

The paper is structured as follows. The next section outlines the background and related work where the decline in Software Engineering popularity is observed and three related literature reviews are used as a basis for the need for additional study. The subsequent section focuses on surveying relevant studies using a mixed method, enriched by Forward and Backward snowballing. Then, a total of 117 studies are reviewed and clustered to gain an insight. A section with conclusions and 27 references closes this study.

II. Background and Related Work

Product and Project management frameworks have matured to deliver expected results based on input data (e.g., architecture, technology, operational model, sprint and release cadence, and DevOps practices). However, the decisions on the input data often are made by human expertise based on empirical experience and available resources. There is rather little practical application of scientific research on tuning the

* Corresponding author's e-mail: Uldis.Karlovs-Karlovskis@rtu.lv
Article received 2024-04-07; accepted 2024-07-12

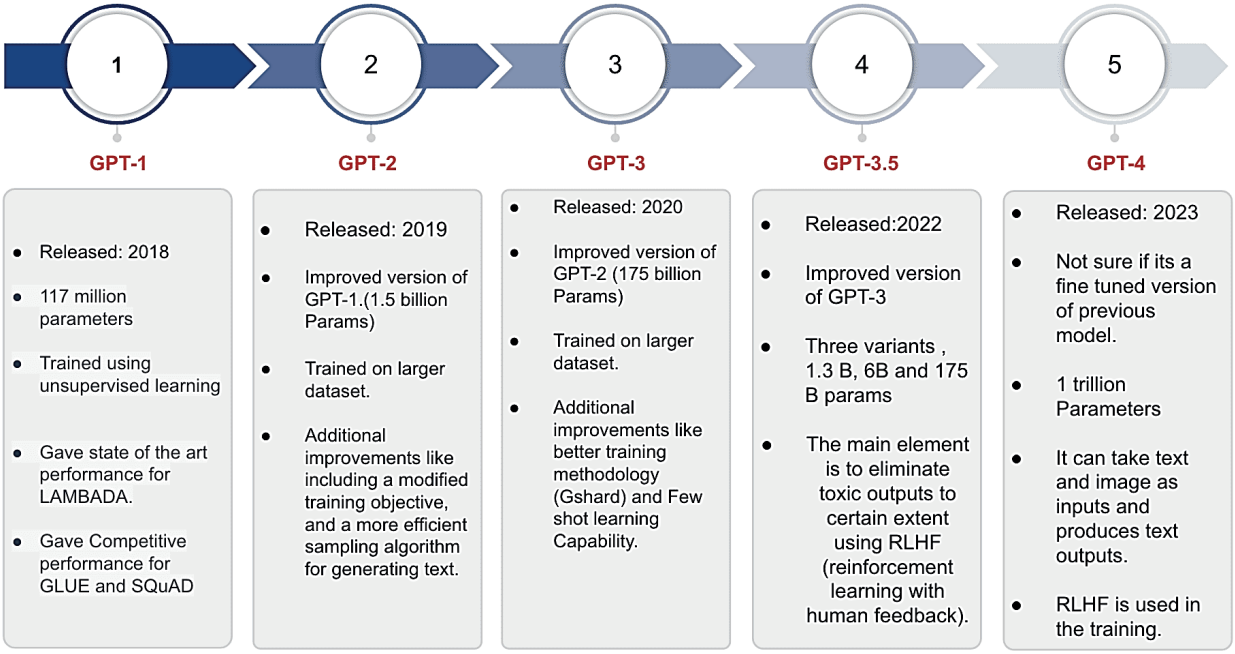

Fig. 1. Evolution of GPT Models [2].

optimal input parameters that optimise the outcomes of organisational performance. The lack of practical application could be explained by reasoning that there are too many input parameters for a human being to process for a pragmatic and sensible decision-making process. The latest developments in Generative AI (based on GPT evolution, see Fig. 1) technology could be able to overcome this issue and help human beings work with the required optimal amount of input parameters.

A thorough research of the latest Generative AI capabilities is required to understand and experiment with its applications in the Software Engineering process and practice. It is a fast-developing field in IT, which means that some identified gaps in the technology could be resolved in less than 6 months. Therefore, it is important to know who the active researchers are, what problems they are working on, and whether there is interest in cooperation or joint work on a particular topic. Such an approach of continuous communication will enable updated knowledge of researchers on the topic going forward.

The goal of this systematic literature review is to understand existing research and applications of Generative AI in software engineering, development, and maintenance. This paper, by focusing on Generative AI capabilities for engineering (see Fig. 2), takes a more focused view than G. Giray [1] has done in the research with "ML for SE". As he explains:

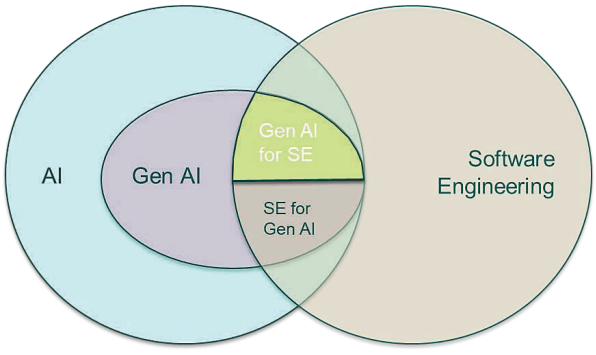

Fig. 2. The highlighted area is the focus of this review illustrated on a Venn diagram.

"ML for SE refers to applying or adapting AI technologies to address various SE tasks, such as software fault prediction, code smell detection, reusability metrics prediction, and cost estimation, etc. Researchers utilize ML models obtained from SE data (source code, requirement specifications, test cases, etc.) to engineer software more efficiently and effectively."

Additionally, this study focuses on Generative AI applications and should help identify ongoing research and researchers in the field of interest for cooperation. The existing and near-future findings in the literature will be used to define a more concrete goal of further research which is scoped by a specific timeframe and resources. Therefore, the following questions are asked:

- RQ1: Which Software Engineering sub-fields are actively experimented on with Generative AI and which ones are underrepresented?
- RQ2: Who are the active researchers in the field for future collaboration?
- RQ3: What are the most common research methods used in the field of Generative AI for Software Engineering research?

Before starting the survey, an analysis of related work found three somewhat similar and recent literature reviews by other authors:

- The term "LLM-based Software Engineering" was coined by researchers [3], which improves the name "ML for SE" – the goal of this review. This literature review concludes that the majority of ongoing research is in the code generation technology with 15 models being highlighted.

Surprisingly, the authors highlight that Requirements Engineering, Design and Refactoring are under-represented in the literature, but ignore the big gap in Delivery, Operations and Virtual Infrastructure domains.

- In a recent review Nishant A. Parikh [4] investigates Generative AI applicability in Software Product Management. There is no comparison for where the majority of research effort is spent, but it is identified that code generation is significant and other areas have lots of potential for further scientific work and innovation.
- Jiachi Chen and a team of researchers selected 123 studies of the year 2023 to understand the current levels of LLMs integrations with seven major tasks of Software Engineering [5] – Code Generation, Code Summarisation, Code Translation, Vulnerability Detection, Code Evaluation, Code Management and Q&A Interaction. As a result of the review, they identified that code and high-coverage test case generation were the hottest topics of the research (see Fig. 3). The authors do recognise Maintenance as an important Software Engineering component but the provided review of Code Maintenance is limited to Version Control, branching and merging of code lines.

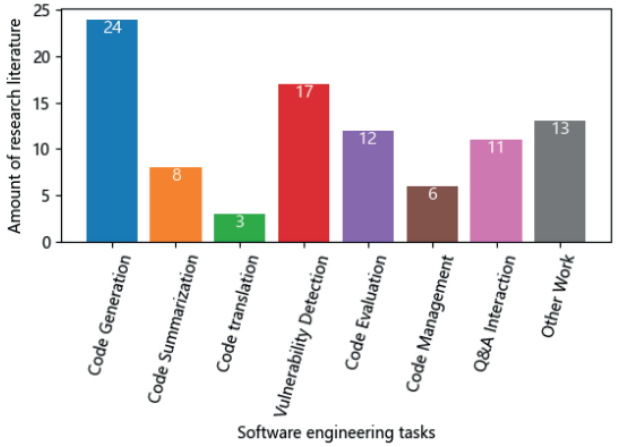

Fig. 3. Number of literature on different Software Engineering tasks [5].

To answer RQ3, as the basis for research methods the framework applied by Kitchenham [6] will be used. It presents seven different research methods: Experiment, Quasi-Experiment, Lessons learnt, Case study, Opinion survey, Tertiary study and Other. Tertiary study will be excluded from the pool due to not being common in the SE scientific studies.

These findings, from existing related work in literature reviews and seemingly declining research in Software Engineering and growing interest in code generation using Artificial Intelligence, require a tailored systematic literature survey to be conducted.

III. Systematic Literature Survey

The systematic literature review is executed using a mixed method. The initial pool of the literature will be coined by: 1) using search keywords of one knowledge search database; and 2) selecting three popular authors of the field and including their publications over the last three closed years starting from the year 2020. In the year 2020, the OpenAI released open API for developers to access AI models, which can be marked as a symbolic beginning of usable Generative AI [7].

To ensure the review of only high-quality literature, the following parameters are applied to the search queries:

- Scopus.com as the source database;
- Range: 2020–2023 as the year of publication;
- Language: English;
- Subject area: Computer science.

The goal of this review is to get a well-covered understanding of the latest achievements and research in Generative AI applicability in Software Engineering work. Therefore, for the database search method a special search string with synonyms for keywords is designed:

(LANGUAGE (english) AND TITLE-ABS-KEY (generative OR gen OR gan) AND TITLE-ABS-KEY (ai OR (artificial AND intelligence) OR (machine AND learning)) AND TITLE-ABS-KEY ((software AND development) OR devops OR agile OR (code AND generation))) AND PUBYEAR > 2019 AND PUBYEAR < 2024 AND (LIMIT-TO (SUBJAREA , "COMP"))

The search string is used in the Scopus database search as a query and returns 246 results where many of which seem to be rather general and out of the scope of the research. Therefore, it is decided to narrow down the search results and focus on reviewing papers about “software engineering” specifically. The following query returns 90 results and further is used in this research:

(LANGUAGE (english) AND TITLE-ABS-KEY (generative OR gen OR gan) AND TITLE-ABS-KEY (ai OR (artificial AND intelligence) OR (machine AND learning)) AND TITLE-ABS-KEY ((software AND engineering))) AND PUBYEAR > 2019 AND PUBYEAR < 2024 AND (LIMIT-TO (SUBJAREA , "COMP"))

To enrich the research pool with additional results that may have been missed using the search query, the manual author-centric search method is applied to find additional papers. The following list of popular figures from the industry gets queried one by one in the Scopus database system: John Allspaw, Patrick Debois, Jez Humble, Nicole Forsgren, Martin Fowler, Gene Kim and David Farley. These researchers and opinion leaders are highly recognised in the DevOps community but in recent years only PhD Nicole Forsgren has published papers that are in the scope of this research. At this point of the review, no researchers working specifically in the target field are known to the researcher; therefore, additionally two unknown but referenced by N. Forsgren’s work researchers are selected, who have published studies related to developer experience or DevEx [8]. As a result, the following authors and their publications starting from 2020 are added to the pool of literature that will be analysed:

- Nicole Forsgren is the author of the book “Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations”. Nicole Forsgren has been researching topics around

development and organisational productivity for more than ten years. In 2020, Forsgren was hired by GitHub as the VP of Research and Strategy. She served as Director of Organisational Performance and Analytics at Chef Software from 2014 to 2015. Forsgren was CEO and co-founder of DevOps Research and Assessment LLC (DORA) with Jez Humble and Gene Kim. In 2018, DORA was purchased by Google [9]. Nicole earned her PhD in Management Information Systems from the University of Arizona, and she is a Research Affiliate at Clemson University and Florida International University [10].

- Christian Bird was a co-author with N. Forsgren for the study "Taking Flight with Copilot" [11]. Bird is a researcher at Microsoft Research working on Empirical Software Engineering as part of the Research in Software Engineering RiSE group. He received his Bachelor degree in CS from Brigham Young University and his PhD in Computer Science from UC Davis, studying empirical software engineering under the advisor Prem Devanbu [12].
- Mik Kersten was a co-author with N. Forsgren for the study "DevOps metrics" [13]. Kersten is a Polish-Canadian computer specialist who created and leads the open-source Eclipse Mylyn project. Kersten invented the Task-Focused Interface technology underlying Mylyn while working on his PhD at the University of British Columbia in Vancouver, British Columbia, Canada. While completing his PhD, Kersten and his PhD supervisor, Gail C. Murphy, founded Tasktop Technologies, which provided productivity software built on the Mylyn technology, but now focuses on providing Value stream management software around Kersten's book "Project to Product" [14].

To select only the studies that are in focus of this research, a clean-up of three steps is conducted. The first clean-up step starts with a pool of 119 primary studies. Before starting to filter the literature pool by abstract, the metadata of each study is listed in one spreadsheet to identify and remove duplicates. In addition, 15 findings appear to be study collections of conferences without any topics identified that may fit the research target. A collection of a recent ACM International Conference Proceeding Series, PROMISE 2023, seems relevant and the full text has been ordered but it has not arrived by finishing the work on this paper. Six findings are identified as duplicates of an existing record. The resulting number of studies is 96. As the second step, the resulting literature is reviewed and sorted by publication titles and abstracts into two categories: 1) research about engineering Generative AI; and 2) research about Generative AI applicability in engineering. Only the latter pool is used going forward because that is the scope of this research. 46 studies are identified as irrelevant. As the third step of the clean-up, for all titles of studies that seem to be ambiguous, the abstract is analysed to identify unrelated studies. 25 more studies are identified as irrelevant. A full text of another three studies is analysed to determine their relevance to the review topic, from which all three get excluded from the primary pool. As a result, the primary pool consists of 22 studies.

To enrich the primary pool with studies that may be relevant to the review but are using keywords that have been missed in the database search query, two more steps are executed:

1. Forward snowball the cited papers and identify papers that fit the review goal;
2. Backward snowball papers that cite the filtered papers and identify papers that fit the review goal.

As the next step of the survey, the forward snowball method is applied. Forward snowball method adds studies that are referenced from the primary pool. Each referenced study is analysed in the same manner as it was done with each study of the primary pool. The time range of this review is four years or all studies starting from the year 2020, therefore, only referenced studies of this period are eligible. The tool "Litmaps" is used to identify referenced studies. The full pool of 3-year forward snowball references consists of several hundred studies, often cross-referencing each other. After de-duplication and filtering the pool by the title of the study, 60 studies have been added to the review pool.

As the last step of gathering the literature list for the survey, the backward snowball method is applied. Backward snowball method adds studies that cite studies from the primary pool. Each new study is analysed in the same manner as it was done with each study of the primary pool. The tool "Litmaps" is used to identify cited-by studies. The full pool of backward snowball studies consists of 171 studies, often cross-referencing the previously found studies or each other. After de-duplication and filtering the pool by titles of studies, 35 additional studies are added to the review pool.

As a result, the final list consists of 117 papers (see Fig. 4) to be reviewed. That is approximately 20 % of the total amount of papers found during the survey and gives enough confidence that the selected list for review covers most of the ongoing research in the field.

IV. Systematic Literature Review

The review starts with 117 thoroughly selected papers (see Fig. 4) from the systematic literature survey and continues with the review structured in the following phases:

- Identify the target Software Engineering sub-field of each study and research the most promising studies which will help answer RQ1;
- Identify the research method of each study for a particular Software Engineering sub-field to answer RQ3;
- To answer RQ2, conduct a visual review of the resulting authors, titles of studies and abstracts to make additional conclusions about the review – identify common themes, authors and keywords that could suggest an extended review of the literature for future research.

To identify the target Software Engineering sub-field of each study, a clustering model is required. To have similar weights of classes on the software operational side compared to product development (to identify underrepresented sub-fields), for the start, the DevOps life cycle illustration (see Fig. 5) of eight phases is chosen over the conventional view of Software

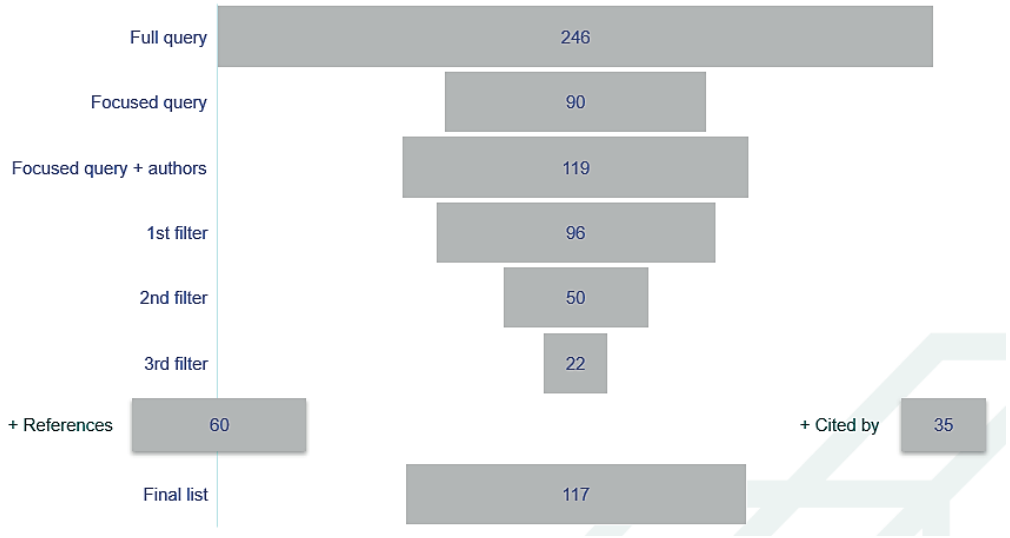

Fig. 4. The highlighted area is the focus of this review illustrated on a Venn diagram.

Delivery Life Cycle's six phases (Planning, Analysis, Design, Implementation, Testing & Integration and Maintenance). In addition to the eight phases, the following classes are added to differentiate some studies and avoid misleading interpretations of results – Documentation, Reverse engineering, Design and Architecture.

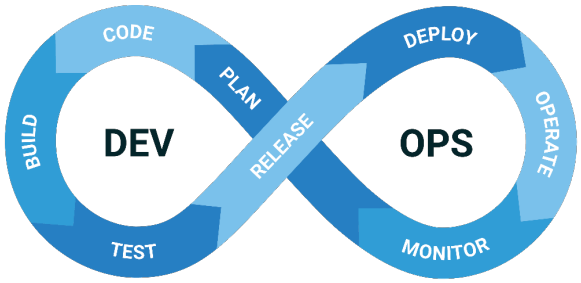

Fig. 5. A high-level visualisation of the DevOps life cycle phases.

Answering RQ3 requires the clusterisation of all studies in different types of research methods. Therefore, the research methods by Kitchenham [6] are used and enriched with one more method "Product study" to separate general and industrial studies for denoising the result. Each research method can be understood in the following way:

- Experiment – an experiment in a laboratory to observe cause and effect. This method often includes the development of some technological tool;
- Quasi-Experiment – a non-randomized experiment in a laboratory to observe cause and effect. This method is rarely used in the Generative AI application due to its nondeterministic behaviour by design;
- Case study – observation of cause and effect in the field;
- Lessons learnt – papers that typically have a broad scope and analyse correlations of more than one use case based on empirical data;
- Opinion survey – qualitative expert interview and quantitative survey analysis;
- Other – the rest of the studies that do not fit the above criteria.
- Product study – research conducted towards a specific technology, usually a commercial product by some vendor, instead of a generally available artefact like an LLM model.

Each title of the study and abstract, when the title is ambiguous, was reviewed to classify each study and research method accordingly. The identified classes of each paper were logged in a spreadsheet for calculation and visual analysis of the results. The review results show the sub-field "Code" as the most popular and that naming a research method generally is not a popular practice. Remaining of the section discusses in detail the findings of the review.

At first, the review result shows a concerning picture – the pie chart visualisation of 117 studies (see Fig. 6) shows that 65 % of studies are focused on code generation using Generative AI tools. It could be explained in a way that this class includes code generators, coding assistants, automated bug-fixing experiments etc. which makes it worthwhile to break the "Code" class into sub-classes in further research. There is also a theoretical probability that an imbalanced proportion of studies was added or missed during the forward and backward snowballing, somehow giving a drift of search results on code generation challenges particularly. Although this theory is disregarded by the three [3], [4] and [5] similar recent literature reviews discussed in the previous section.

The findings of the current review are reported in Table II. It shows a concerning trend – a rather little effort (35 % in total) is being spent on the other Software Engineering phases before and after the "Code" phase. This analysis generally answers RQ1 and reconfirms that there is a major gap in the research of Generative AI for non-coding SE areas and a lot of room for further research. The motivation for asking RQ3 was to understand the most popular methods among researchers when performing their studies lately – how often research methods are identified and in what manner.

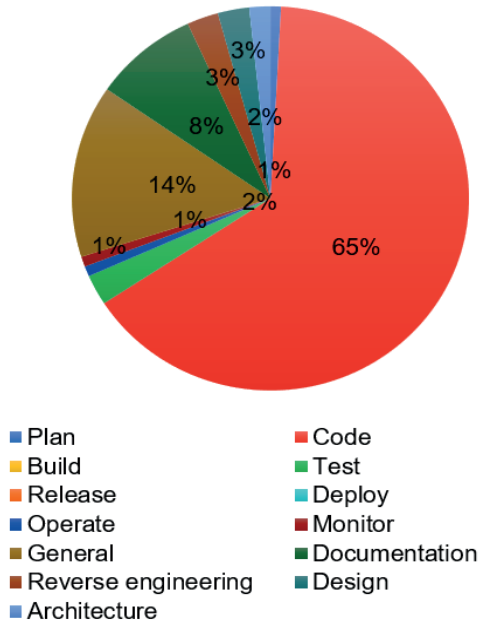

Fig. 6. A pie chart of sorted 117 studies.

Figure 7 visualises the results as a pie chart and shows that the research method "Experiment" is used in the majority (55 %) of studies. A rather big portion (19 %) of studies is based on a particular technology, the product studies. The third biggest group is opinion surveys which consist of social surveys, expert interviews, theoretical essays etc. However, it is worth noting that the research method rarely is clearly stated in the paper and these conclusions are based on the author's assumptions and expertise.

Further, to gain an insight into the latest research in each SE sub-area studies of the highest relevance (based on the title, abstract or the number of *cited-by* studies) to the research topic, a full-text read is done for at least one study of each phase. First, the cluster of more general studies is analysed, which is continued by order of software development lifecycles and finished with the unconventional cluster of experiments in reverse engineering.

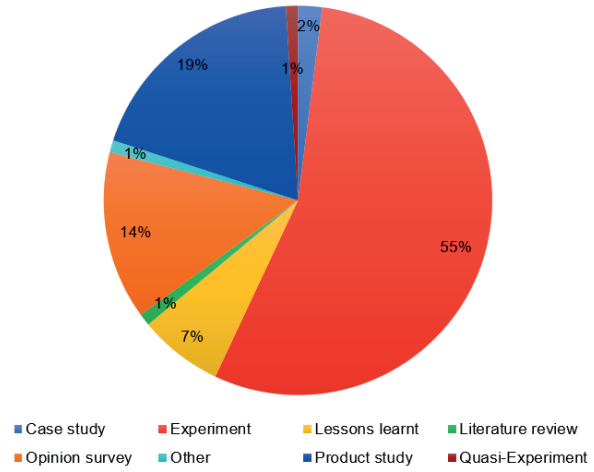

Fig. 7. A pie chart of classified research methods by assumption.

The 17 general studies of the survey pool are mostly opinion surveys to research impact on the society, usually engineers, who are or will be using Generative AI in their work. The results usually show that engineers are willing to accept the new technology but the required organisational changes open new challenges to solve. As was already identified in the study by Jiachi Chen [5], the general focus of the latest research is on various code generation tasks for different problems, and for now, succeeding only in context-unaware scenarios. This pattern could be caused by the reality that researchers can efficiently experiment only with public databases, which typically contain only the open-source code. Four of the studies are literature reviews. Some studies (e.g., [15], [16]) are rather opinionated articles without deep research to actualise what generally the AI buzz is about.

The first SE sub-field is plan with only one study [17]. The study mentions a few other similar studies from earlier days but claims that in 2023 this was the first application of GPT-2 Transformer-type architecture for estimating Story Points. The research additionally concludes that people appreciate explainable AI-based techniques over the black box. While

TABLE II

MAPPING OF REVIEWED SE SUB-FIELDS AND RESEARCH METHODS

Sub-fields/ methods	Case study	Experiment	Lessons learnt	Literature review	Opinion survey	Product study	Quasi-Experiment	Other	Total
General		1		4	10	1		1	17
Plan		1							1
Architecture					1	1			2
Design		1	1			1			3
Code		42	5		13	15		1	76
Build									0
Test		2		1					3
Release									0
Deploy									0
Operate							1		1
Monitor						1			1
Documentation	2	6	1			1			10
Reverse engineering		3							3

story point estimation in the Scrum methodology may seem like a routine task to automate, the teams must not forget the social and learning components during the Poker Planning sessions.

The architecture sub-field, where one [18] of the 2 studies has been full-text read, is chosen as the next logical phase. Researchers of this study experimented with ChatGPT to help architects with their work. The second study is an opinion survey. Such a small number of studies show that generating system architecture or applying model-driven approaches using Generative Artificial Intelligence are highly underrepresented fields.

The design sub-field of this research is targeting the visual and user interface creation tasks of the Software Engineer which results in three studies in total. The full text read [19] study has a major focus on investigating and assessing a Generative AI-based tool named Instigator. The authors successfully generate GUI layouts of different depths by providing clear instructions as text to the tool. The study mentions 3–5 other tools designed for similar purposes. At this point, it is unclear how many of those are commercial or scientific work. While making the machine draw UIs sounds like a remarkable result, it is truly useful only for the first wireframe and then a professional designer must take over the changes. A model-driven approach using Generative AI that can maintain several interfaces at the same time would be a promising area for further research.

As it was previously identified, the Code sub-field in Software Engineering of 76 studies has been the most popular among researchers when experimenting with AI. Not all of those are using Generative AI technology. 34 of those studies are surveys or usability experiments of some existing technology, but the remaining 42 seem to be novel and push-up abilities of Artificial Intelligence. The study [20] is an experiment to develop a code merge tool that uses deep learning, not Generative AI technology. The study [21] promises advancement in Software Security when it focuses on memory safety bugs in C particularly. The research is motivated by the finding that gpt-3.5-turbo has a 41.3 % chance to generate code with buffer overflow issues and proposes a novel approach to use a rule-based system in combination with Generative AI sending results iteratively to each other. This is an actively researched field in several directions where researchers probably should collaborate more to achieve the results faster.

Not a single study has been identified for the Build sub-field. Studies mention DevOps but those were a bitter fit in the General class. The potential of future research is unknown and therefore hypothetical.

The literature review of [22] focuses on Artificial Intelligence used for the Test sub-field and covers 52 relevant studies since 2020 of which about a half are related to code generation (see Fig. 8). Also, there are experiments to generate test scenarios, process test results etc. with novel results. However, the authors identify that there is a significant gap in research for Integration and Acceptance test generation.

Similarly, to the Build sub-field, also for the Release and Deploy sub-fields, no studies have been identified researching applications of Generative AI. If for Release this could be explained with a reasoning that release activity generally is less frequent than other activities, this is hardly applicable to the Deploy phase, although that does not make them less valuable to research. There must be enormous potential for future research and novelty.

One study for the Operate sub-field has been identified. After a full text read the study from 2021 [23] was identified as an experiment of deep learning application where no Generative AI technology had been applied. A model-driven approach coupled with Generative AI capabilities could open new areas of future research.

For the Monitor sub-field, the only identified study [24] in the log processing area confirms that not much other study exists. Since the study is experimenting with OpenAI ChatGPT 3.5-turbo service, it is classified as a “Product study”. It exercises exploratory experiments of use cases and makes conclusions about the results. The future research using Large Language Models to analyse log contents opens a major potential for researchers.

Documentation in software projects has always been an under-delivered area. Therefore, it is not surprising that 10 studies are dedicated to solving the often-social problem in Software Engineering. A promising study [25] about code comment generation gives a good background on the latest challenges and a strong comparison with three similar tools. The latest state-of-the-art tools can generate comment from a simple code, although there is no deeper understanding of the program and that remains a challenge in all studies. The Generative AI for now is context-unaware and cannot work with an environment of many abstraction layers at the same time like humans.

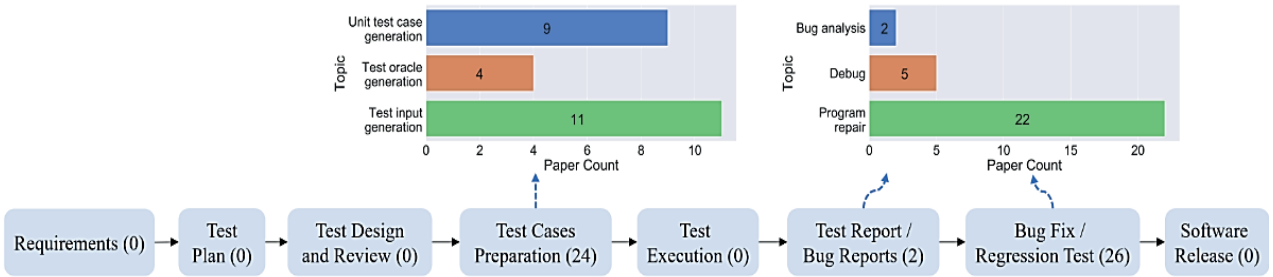

Fig. 8. Distribution of testing tasks with LLMs [22].

The last sub-field and in a way the most exciting is the ongoing research in Reverse Engineering. Scientists have been looking for ways how to decompile readable code from binary programs since the beginning of programming. All three studies focus on ready binaries and reading compiled code. The authors of Nova+ [26] state that it is the first and only Generative AI for binary code. In the future, such a novel solution could potentially translate X86-64 binary code to ARM64, which would save a lot of ongoing effort to support both architectures that need to be compiled separately.

The third (RQ3) research question was "What are the most common research methods used in the field of Generative AI for Software Engineering research?" To get a statistical view of which research methods are more popular, for all of the studies the titles and abstracts were analysed and an assumption was made about the used method for each study from the pool. When general studies are excluded, which gives 100 relevant studies in total, Fig. 7 visualises the majority of the research method as experiment (55 %); product study (19%) and opinion survey (14%) as the second and third most popular methods. Since Generative AI generally is non-deterministic, quasi-experiment research method requirements are hard to achieve making it the least used method. A common approach is to perform an experiment and afterwards often survey experts

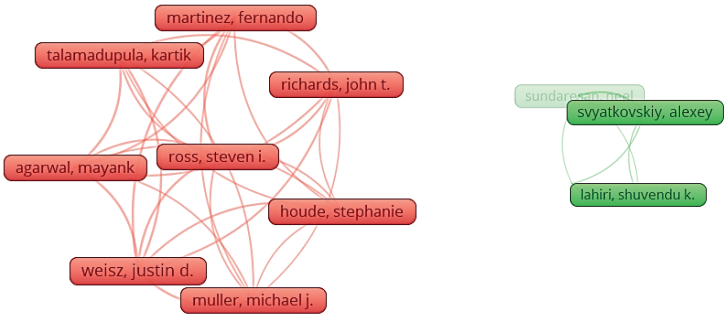

Fig. 9. Association analysis of the most productive authors.

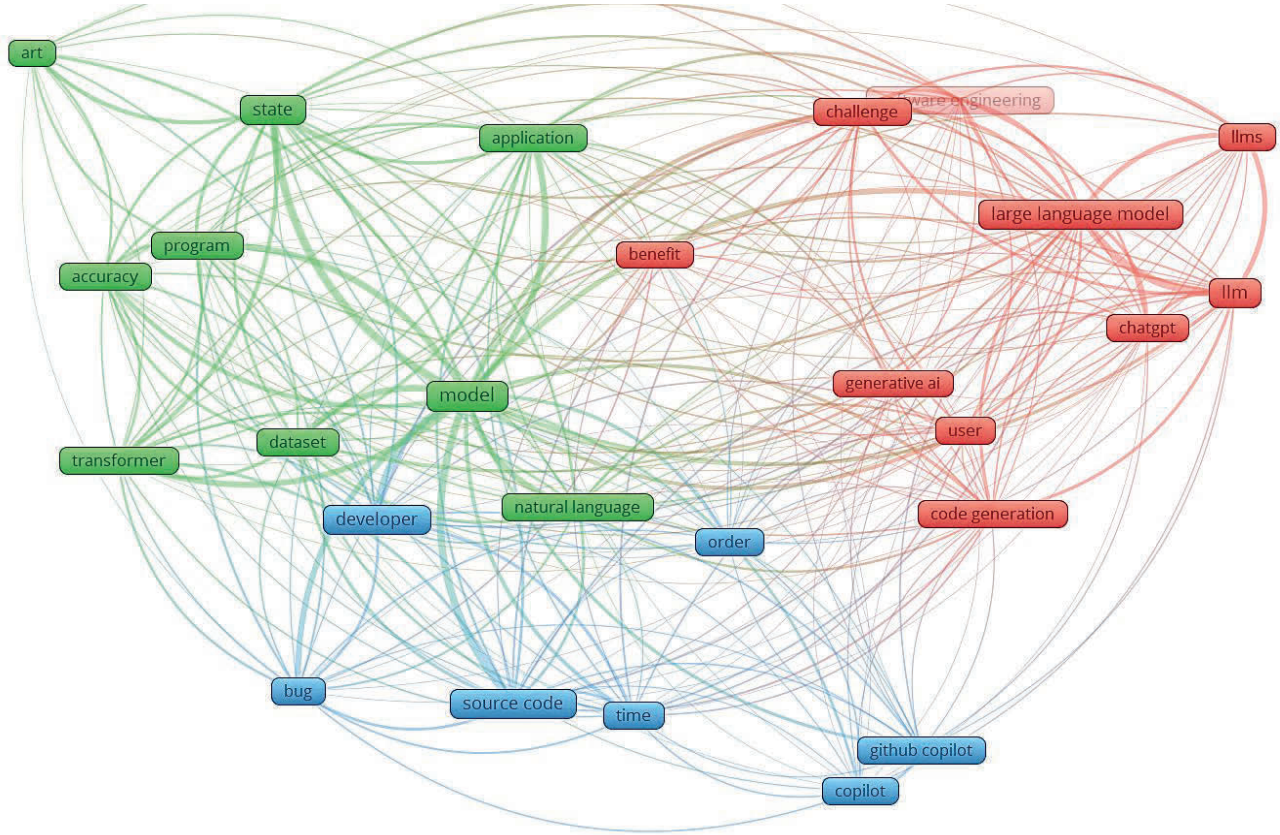

Fig. 10. Network visualisation of keywords from titles of studies and abstracts.

validate the results. Even though, more than 90 % of the research can be easily disqualified [3] when standard evaluation quality constraints are applied. It is worth noting that there were close to zero case studies identified and the lessons learnt often were rather general observations. This shows the immaturity of the research field and the large potential for future work.

The second (RQ2) research question was "Who are the active researchers in the field for future collaboration?" The visual analysis is performed with the tool VOSviewer on the 117 studies that were reviewed. The result (see Fig. 9) shows that there are two "camps" of authors who have been contributing the most. A deeper analysis of expertise and target research by these authors would be required to make any further conclusions. As Fig. 9 visualises, about 12 different authors have been identified as the most productive. The total pool of authors counted was 432. The research scope has been too wide to select anyone specific for further analysis at this point.

As an additional experiment, the VOSviewer visualisation is also used to create a Network Visualisation of keywords from titles of studies and abstracts (see Fig. 10). From the output diagram, it has been observed that "llm", "llms" and "large language model" are almost like synonyms to use in the future database search queries. When searching for studies on Generative AI, the keywords "model" and "transformer" are also worth considering after the OR operator.

Via reviewing 117 studies, RQ1 of this study has been answered up to a satisfying level to gain insight into which Software Engineering sub-fields are actively experimented on with Generative AI and which ones are underrepresented. Also, RQ2 and RQ3 have been answered on a general level but for each SE sub-field differences are expected and can be researched further.

V. Conclusions and Future Research

This paper focuses on reviewing the use of Generative Artificial Intelligence for Software Engineering. At the beginning of the research, we asked which Software Engineering sub-fields are actively experimented on and which are underrepresented. A clear trend in code generation tasks is visible in the current state of Generative AI for Software Engineering. First, researchers are conducting experiments to advance the existing technology by combining its components. Second, the research is ongoing for testing capabilities by using the latest technology like ChatGPT and GitHub Copilot. Two interconnected groups of researchers have been identified but formally stating a research method in the study is not a popular practice.

As for the gaps in the latest research, it has been observed that case studies are rare to find which means that scientists are making novel artefacts which do not get implemented in the field. The proportionally small number of studies in SE sub-fields like Build, Release, Deploy, Operate and Monitor indicates a broad untapped area for cost savings and novelty, which could bring major optimisation in Software Engineering generally. This could be explained by the paradox that expertise in these topics typically is gained in the field where scaling and complexity are a challenge, but academic researchers typically don ot get much exposure to this problem area. A granular and tailored systematic literature review with a focus on Generative AI for "the right side" of Software Engineering (Build, Release, Deploy, Operate and Monitor) is suggested.

One of the potential areas of future research would be applying model-driven engineering principles and AI algorithms to create a solution that automates process and artefact generation for Build, Release and Deploy sub-phases of Software Engineering, a DevOps Automation Solution.

References

[1] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *Journal of Systems and Software*, vol. 180, Oct. 2021, Art. no. 111031. https://doi.org/10.1016/j.jss.2021.111031

[2] A. Mehra, M. Yadav, and B. Ammu, "GPT history." [Online]. Available: https://www.kdnuggets.com/2023/05/deep-dive-gpt-models.html. Accessed on: Jan. 03, 2024.

[3] A. Fan *et al*., "Large language models for software engineering: Survey and open problems," *arXiv.org*, Oct. 2023. https://doi.org/10.48550/ARXIV.2310.03533

[4] N. A. Parikh, "Empowering business transformation: The positive impact and ethical considerations of Generative AI in software product management – A systematic literature review," *arXiv.org*, Jun. 2023. https://doi.org/10.48550/ARXIV.2306.04605

[5] Z. Zheng *et al*., "Towards an understanding of large language models in software engineering tasks," *arXiv:2308.11396*, Aug. 2023. https://doi.org/10.48550/arXiv.2308.11396

[6] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Inf. Softw. Technol.*, vol. 55, no. 12, pp. 2049–2075, Dec. 2013. https://doi.org/10.1016/j.infsof.2013.07.010

[7] "OpenAI official web site." [Online]. Available: https://openai.com/blog/openai-api. Accessed: Dec. 26, 2023.

[8] A. Noda, M. Storey, N. Forsgren, and M. Greiler, "DevEX: What actually drives productivity?" *Communications of the ACM*, vol. 66, no. 11, pp. 44–49, Oct. 2023. https://doi.org/10.1145/3610285

[9] "Nicole Forsgren biography." [Online]. Available: https://en.wikipedia.org/wiki/Nicole_Forsgren. Accessed: Dec. 26, 2023.

[10] "Nicole Forsgren official web site." [Online]. Available: https://nicolefv.com/. Accessed on: Dec. 26, 2023.

[11] C. Bird *et al*., "Taking flight with Copilot," *Communications of the ACM*, vol. 66, no. 6, pp. 56–62, May 2023. https://doi.org/10.1145/3589996

[12] "Microsoft Research web site – Bird." [Online]. Available: https://www.microsoft.com/en-us/research/people/cbird/. Accessed on: Dec. 26, 2023.

[13] N. Forsgren and M. Kersten, "DevOps metrics," *Communications of the ACM*, vol. 61, no. 4, pp. 44–48, Mar. 2018. https://doi.org/10.1145/3159169

[14] "Mik Kersten biography." [Online]. Available: https://en.wikipedia.org/wiki/Mik_Kersten. Accessed on: Dec. 26, 2023.

[15] I. Ozkaya, "The next frontier in software development: AI-augmented software development processes," *IEEE Software*, vol. 40, no. 4, 2023. https://doi.org/10.1109/MS.2023.3278056

[16] C. Ebert, P. Louridas, and C. Ebert, "Generative AI for Software Practitioners," IEEE Softw, pp. 4–9, July-Aug. 2023. https://doi.org/10.1109/MS.2023.3265877

[17] M. Fu and C. Tantithamthavorn, "GPT2SP: A transformer-based agile story point estimation approach," *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 611–625, Feb. 2023. https://doi.org/10.1109/TSE.2022.3158252

[18] I. Ozkaya, "Can Architecture Knowledge Guide Software Development With Generative AI?, " *IEEE Software*, Sep.-Oct. 2023. https://doi.org/10.1109/MS.2023.3306641

[19] P. Brie, N. Burny, A. Sluÿters, and J. Vanderdonckt, "Evaluating a large language model on searching for GUI layouts," *Proceedings of the ACM on Human-Computer Interaction*, vol. 7, no. EICS, Jun. 2023, Art. no. 178. https://doi.org/10.1145/3593230

[20] E. Dinella, T. Mytkowicz, A. Svyatkovskiy, C. Bird, M. Naik, and S. K. Lahiri, "DeepMerge: Learning to merge programs," *IEEE Transactions on Software Engineering*, *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1599–1614, April 2021. https://doi.org/10.1109/TSE.2022.3183955
[21] Y. Charalambous, N. Tihanyi, R. Jain, Y. Sun, M. Ferrag, and L. Cordeiro, "A new era in software security: Towards self-healing software via large language models and formal verification," *arXiv.org*, May 2023. https://doi.org/10.48550/ARXIV.2305.14752
[22] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language model: Survey, landscape, and vision," *arXiv.org*, Jul. 2023. https://doi.org/10.48550/ARXIV.2307.07221
[23] M. Shetty, C. Bansal, S. Kumar, N. Rao, N. Nagappan, and T. Zimmermann, "Neural knowledge extraction from cloud service incidents," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Madrid, Spain, May 2021, pp. 218–227. https://doi.org/10.1109/ICSE-SEIP52600.2021.00031
[24] P. Mudgal and R. Wouhaybi, "An assessment of ChatGPT on log data," *arXiv.org*, Sep. 2023. https://doi.org/10.48550/ARXIV.2309.07938
[25] Y. Park, A. Park, and C. Kim, "ALSI-transformer: Transformer-based code comment generation with aligned lexical and syntactic information," *IEEE Access*, vol. 11, pp. 39037–39047, Apr. 2023. https://doi.org/10.1109/ACCESS.2023.3268638
[26] N. Jiang, C. Wang, K. Liu, X. Xu, L. Tan, and X. Zhang, "Nova+: Generative language models for binaries," *arXiv.org*, Nov. 2023. https://doi.org/10.48550/ARXIV.2311.13721

Uldis Karlovs-Karlovskis is a PhD student at Riga Technical University, Latvia. His current research focus is DevOps practice automation using artificial intelligence and model-driven software development techniques.
E-mail: Uldis.Karlovs-Karlovskis@rtu.lv
ORCID iD: https://orcid.org/0009-0008-7816-3770