Generative AI in Software Engineering: A Systematic Mapping Study based on SWEBOK Areas

Carlos Pulido[0009-0008-8122-3500], Mª Ángeles Moraga[0000-0001-9165-7144], Félix García[0000-0001-6460-0353], and Coral Calero[0000-0003-0728-4176]

Instituto de Tecnologías y Sistemas de Información, Camino de Moledores, s/n, Ciudad Real, Castilla-La Mancha, 13005, Spain

Abstract. Traditionally, software development has been predominantly supported by a variety of non-AI tools and methodologies. However, with the introduction of Generative AI (GenAI) tools such as ChatGPT, the landscape has changed significantly, opening up new opportunities to improve all activities that are part of the software development process. To explore the usage of GenAI in different areas of Software Engineering (SE) as defined by the SWEBOK: *Requirements*, *Architecture and Design*, *Construction*, *Testing*, and *Maintenance*, a Systematic Mapping Study (SMS) was conducted using the SCOPUS database, obtaining a total of 256 studies, of which 48 were selected as primary studies. The results indicate that GenAI is most widely used in the areas of *Construction* and *Testing*, improving efficiency by automating repetitive and structured tasks. In contrast, areas that require more creativity and strategic thinking pose significant challenges for the evolution of GenAI, where improving the synergy between GenAI and human experience is essential. Therefore, it is important that future efforts focus on evolving GenAI capabilities to better address these gaps and expand its role throughout the SE discipline.

Keywords: Generative AI (GenAI), Software Engineering (SE), Systematic Mapping Study (SMS)

A Methodology

This appendix describes the steps taken in planning and conducting this SMS, for which the guidelines by [34] were followed.

A.1 Research objective and question

The aim of this review is to identify the specific areas of the SWEBOK that are most affected by the paradigm shift brought about by the use of GenAI. To guide the study, we adopted the Goal-Question-Metric (GQM) model [42]. The first step is to define the goals of the study, which are then mapped to specific questions. This ensures that the metrics are aligned with the research objectives and lead to meaningful findings.

The following are the goals (G) and questions (Q) formulated for this study:

G1. To identify the demographics of the area.
- **Q1.1.** What are the trends in terms of years of publication in the area?
  Understanding publication trends is crucial as it highlights the evolution and relevance of the research topic over time.
- **Q1.2.** What are the most common types of academic publications?
  We are interested in identifying the types of academic publication with the highest number of primary papers in the area, such as articles, conference papers, or books.

G2. To identify the areas of the SWEBOK and specific activities within those areas that are susceptible to changes due to GenAI integration.
- **Q2.1.** Which SWEBOK areas are researchers focusing on?
  Our aim is to understand which areas of the SWEBOK researchers tend to focus on. The SWEBOK knowledge areas [5] serve as a guide to the widely accepted principles in SE, and the focus of this SMS is based on, but not limited to, these areas. It is worth emphasizing that a single study may address multiple SWEBOK areas, particularly those that are closely related and often tackled together, such as *Construction* and *Testing*.
- **Q2.2.** What specific activities are researchers focusing on?
  We aim to explore specific activities involved in integrating GenAI into software development, such as its use in optimizing software design, predicting potential bugs, and other related tasks.

G3. To determine which GenAI models are most frequently used across SE practices.
- **Q3.1.** What are the most commonly used GenAI models?
  We aim to determine whether a single type of GenAI model is widely used across all areas of the SWEBOK, or if specific models are consistently applied to particular areas.

G4. To understand the maturity of research.
- **Q4.1.** What is the contribution facet provided by researchers?
  The contribution facet refers to the novel contributions made by researchers [34], which are classified as approach, model, or tool. An approach describes a method or set of rules for how tasks should be performed. A model provides a description of the real world, omitting certain details, and has a higher degree of formality, including semantics and notations. A tool refers to software developed to support the implementation of a proposed solution.
- **Q4.2.** What is the research strategy used by the researchers providing empirical evidence to support their claims?
  Research takes place in different settings, referring to the environments or contexts in which researchers conduct their studies. We are interested in understanding the research strategies used to support

claims, for which the classification according to [43] is followed: field study, field experiment, experimental simulation, laboratory experiment, judgment study, sample study, formal theory or computer simulation.

These goals and their associated questions will guide the SMS process and will later be reformulated as Research Questions (RQs) to systematically answer the objectives of this study.

A.2 Research process

Figure 1 illustrates the research process. In the first step, an automated search was conducted using the Scopus search engine. A total of 256 relevant studies were initially identified. After applying the inclusion and exclusion criteria, we selected the studies most relevant to the purpose of the SMS, resulting in 48 studies. Finally, a report was produced after the data extraction and synthesis of these studies. The details of each step of our research process are documented below.

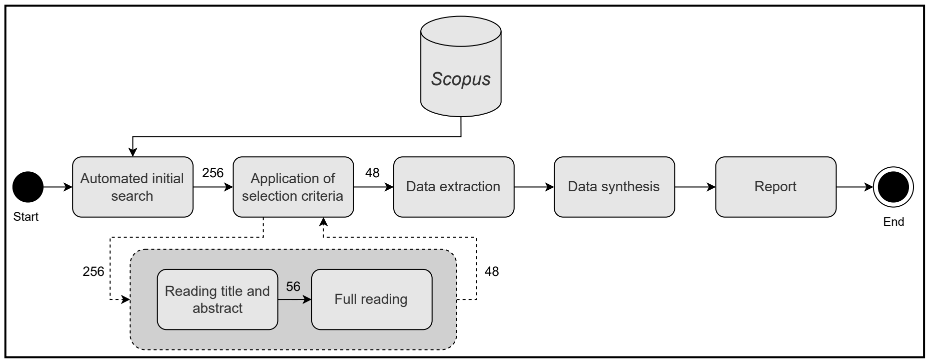

Fig. 1. SMS process overview.

The initial search was conducted systematically using the Scopus database. A targeted search string (see Table 1) was employed to identify relevant research studies related to GenAI and its applications in the SWEBOK areas. To search for relevant literature, following the guidelines suggested by [34], we designed a query using PICO (population, intervention, comparison, outcomes) search with two elements. Population (P): to retrieve literature with titles, abstracts or relevant terms related to the SWEBOK areas, identified by the term "software development", and intervention (I): to retrieve literature related to GenAI or related synonyms and acronyms. We did not consider comparison (C) and outcome (O) as these are outside the scope of this SMS.

Regarding the (I) part of the search query, terms such as "Generative AI" and "GenAI" (as an abbreviation) are essential, as they represent the key concepts

Table 1. Automated search query.

TITLE-ABS-KEY ("software development") AND TITLE-ABS-KEY ("automatic programming" OR "large lange model" OR "llm*" OR "AI chatbot" OR "conversational AI" OR "generative AI" OR "GenAI")*

Table 2. Search query structure.

Search query	**Keywords**
Population (P)	*software development*
Intervention (I)	automatic programming *large language model** *llm** *AI chatbot* *conversational AI* *generative AI* *GenAI*

within the scope of this SMS. Terms like "large language model" and "LLM" (as an abbreviation) were also considered necessary, as LLMs are often the core technology behind many GenAI applications. Furthermore, terms such as "AI chatbot" and "conversational AI" were included, as these applications often use the same underlying technologies as GenAI to generate human-like interactions. Terms like "automatic programming" do not necessarily imply the use of GenAI, as they can also include methodologies such as Model-Driven Engineering (MDE) for code generation through models. However, we decided to include this term in the search to avoid missing potentially relevant articles. After applying the inclusion and exclusion criteria, only those studies that were consistent with the aim of the SMS were selected.

The search was conducted for studies published between 1 January 2022 and 31 July 2024, using the above search query, and resulted in the identification of 256 potentially relevant studies. While GenAI began to gain attention in the mid-2010s, particularly with the development of Generative Adversarial Networks (GANs) [16] in 2014, it was not until 2020 that the technology experienced truly massive growth in real-world applications [7]. The GPT-3 model, released in 2020, was a major breakthrough in natural language generation and paved the way for the development of ChatGPT. ChatGPT, launched in 2022, further expanded the visibility of GenAI and led to its widespread adoption across multiple industries. This surge in interest and application of GenAI, particularly from 2022 forward, makes this period crucial for our research.

After the initial identification of potentially relevant studies from the search engine, we selected those that best met the objective of the SMS, based on the inclusion and exclusion criteria. A study is considered included if it meets all the inclusion criteria and does not meet any of the exclusion criteria. Table 3 shows the Inclusion (I) and Exclusion (E) criteria used:

Table 3. Inclusion and exclusion criteria.

ID	Description
I1	Studies covering one or more areas of the SWEBOK.
I2	Studies addressing GenAI.
I3	Studies focusing on the integration of GenAI into one or more areas of the SWEBOK.
I4	Recent publications (from 2022 to mid-2024).
E1	Studies that evaluate existing methods without proposing new solutions.
E2	Studies that are systematic reviews, literature reviews or meta-analyses that summarize existing knowledge without introducing new concepts.
E3	Non-English sources.

The inclusion criteria I1-I3 ensure that the studies not only focus on GenAI and encompass one or more areas of the SWEBOK, but also address its integration into software development and explore how software developers can benefit from its application. Criterion I4 is specifically defined to include only recent publications, as the use of GenAI in the SDLC is now considered a more mature trend, supported by recent advancements and widespread adoption [7]. Meanwhile, the exclusion criteria filter out studies that: (E1) do not propose new solutions for automating software development through the use of GenAI, particularly those that focus on evaluations that compare different models; (E2) are systematic reviews, literature reviews or meta-analyses that do not provide original research contributions; and (E3) are non-English sources.

As shown in Figure 1, of the 256 potentially relevant studies, 56 primary studies remained following an initial review of titles and abstracts. After a full reading of the studies, 48 studies remained. Detailed information about this SMS can be found in the supplementary material included in the repository [37].

To achieve the research objective and answer the questions (see Section A.1), we systematically extract data from the final set of selected studies. The data extraction fields used for this SMS are as follows:

- *Title:* the title of the study.
- *Author and index keywords:* keywords used for indexing the study, which help in categorizing and locating the research within data.
- *Type of publication:* the type of publication, indicating where the paper was published.
- *Year of publication:* the year of publication of the study, which is crucial in assessing the timeliness of the research..
- *Abstract:* the abstract of the paper.
- *SWEBOK area:* the area(s) addressed in the study, providing information on where GenAI is applied within the software development process. Based on the SWEBOK [5], the areas are: *Requirements*, A*rchitecture and Design*, *Construction*, *Testing*, and *Maintenance*.
- *Activity:* The specific activity(ies) where GenAI is used within each area of the SWEBOK. These activities are based on current industry practices such

as requirements modeling, design modeling, code documentation, or bug detection.
- *Contribution:* A brief description of the contribution of the study, highlighting what has been done and how it contributes to the problem or area addressed.
- *Contribution facet:* The specific type of contribution made by the study.
- *GenAI Model:* Information on the specific GenAI model(s) used in the research.
- *Research strategy:* The research strategy, if any, used to provide empirical evidence for the proposed solution.

After systematically extracting the relevant data from the included studies, the next step involves organizing and analyzing this information. The data was then tabulated to show:

(i) The number of primary studies published per year, showing the development of interest and activity in this area.
(ii) The number of primary studies by publication type, which provides an overview of the preferred dissemination channels for research in the field.
(iii) The most common terms in the selected studies to show the terminology used in the research landscape.
(iv) The number of studies covering each SWEBOK area, providing insight into the distribution of research focus across different areas.
(v) The target activities within each SWEBOK area that help determine how GenAI is being used within the field.
(vi) The most commonly used GenAI models across SWEBOK areas, highlighting which models are preferred for specific tasks and areas within the software development.
(vii) The contribution facet of the studies, which aims to identify the specific ways in which each study contributes in this area.
(viii) The research strategy used when evaluating solutions for integrating GenAI into the software development, showing different methods for conducting SE research.

B Primary studies

This appendix lists all primary studies found through the SMS search process.

Table 4: List of primary studies

Article ID	**Ref.**	**Title**
[PS1]	[6]	Generating domain models from natural language text using NLP: a benchmark dataset and experimental comparison of tools
[PS2]	[54]	LLM-Based Agents for Automating the Enhancement of User Story Quality: An Early Report

Continued on next page

Table 4 – continued from previous page

Article ID	Ref.	Title
[PS3]	[33]	Transforming Software Requirements into User Stories with GPT-3.5 - An AI-Powered Approach
[PS4]	[19]	AI-guided Model-Driven Embedded Software Engineering
[PS5]	[24]	Toward a Symbiotic Approach Leveraging Generative AI for Model Driven Engineering
[PS6]	[40]	DesignSystemsJS – Building a Design Systems API for Aiding Standardization and AI Integration
[PS7]	[45]	A Composable Just-In-Time Programming Framework with LLMs and FBP
[PS8]	[1]	AI-Powered Code Review Assistant for Streamlining Pull Request Merging
[PS9]	[36]	Aligning Documentation and Q&A Forum through Constrained Decoding with Weak Supervision
[PS10]	[2]	Automatic Code Generation From Low Fidelity Graphical User Interface Sketches Using Deep Learning
[PS11]	[28]	CCTEST: Testing and Repairing Code Completion Systems
[PS12]	[29]	Compositional API Recommendation for Library-Oriented Code Generation
[PS13]	[55]	GAP-Gen: Guided Automatic Python Code Generation
[PS14]	[41]	Git Merge Conflict Resolution Leveraging Strategy Classification and LLMn
[PS15]	[47]	Just-In-Time TODO-Missed Commits Detection
[PS16]	[32]	KubePlaybook: A Repository of Ansible Playbooks for Kubernetes Auto-Remediation with LLMs
[PS17]	[23]	LARCH: Large Language Model-based Automatic Readme Creation with Heuristics
[PS18]	[21]	LLM Security Guard for Code
[PS19]	[3]	MicroRec: Leveraging Large Language Models for Microservice Recommendation
[PS20]	[26]	Object Oriented BDD and Executable Human-Language Module Specification
[PS21]	[52]	Project-specific code summarization with in-context learning
[PS22]	[22]	Software Metadata Classification based on Generative Artificial Intelligence
[PS23]	[51]	UniLog: Automatic Logging via LLM and In-Context Learning
[PS24]	[49]	You Augment Me: Exploring ChatGPT-based Data Augmentation for Semantic Code Search
[PS25]	[46]	Aggregating Industrial Security Findings with Semantic Similarity-Based Techniques
[PS26]	[25]	An Ensemble Method for Bug Triaging using Large Language Models

Continued on next page

Table 4 – continued from previous page

Article ID	Ref.	Title
[PS27]	[35]	Automatic Generation of Test Cases based on Bug Reports: a Feasibility Study with Large Language Models
[PS28]	[38]	CAT-LM Training Language Models on Aligned Code and Tests
[PS29]	[10]	Effective test generation using pre-trained Large Language Models and mutation testing
[PS30]	[17]	Exploring the Potential of Pre-Trained Language Models of Code for Automated Program Repair
[PS31]	[18]	Fault-Aware Neural Code Rankers
[PS32]	[50]	Fuzz4All: Universal Fuzzing with Large Language Models
[PS33]	[53]	Gamma: Revisiting Template-based Automated Program Repair via Mask Prediction
[PS34]	[20]	InferFix: End-to-End Program Repair with LLMs
[PS35]	[14]	MissConf: LLM-Enhanced Reproduction of Configuration-Triggered Bugs
[PS36]	[48]	Parameter-Efficient Multi-classification Software Defect Detection Method Based on Pre-trained LLMs
[PS37]	[31]	A Mixed Reality Approach for Innovative Pair Programming Education with a Conversational AI Virtual Avatar
[PS38]	[4]	AI-Driven Refactoring: A Pipeline for Identifying and Correcting Data Clumps in Git Repositories
[PS39]	[27]	Automated API Docs Generator using Generative AI
[PS40]	[12]	An Approach for Rapid Source Code Development Based on ChatGPT and Prompt Engineering
[PS41]	[13]	CodeFuse-13B: A Pretrained Multi-lingual Code Large Language Model
[PS42]	[9]	Conversational Assistants for Software Development: Integration, Traceability and Coordination
[PS43]	[8]	F-CodeLLM: A Federated Learning Framework for Adapting Large Language Models to Practical Software Development
[PS44]	[11]	Meet C4SE: Your New Collaborator for Software Engineering Tasks
[PS45]	[39]	The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development
[PS46]	[44]	Design and Development of a Log Management System Based on Cloud Native Architecture
[PS47]	[30]	LLMParser: An Exploratory Study on Using Large Language Models for Log Parsing
[PS48]	[15]	X-Lifecycle Learning for Cloud Incident Management using LLMs

Table 5: Primary studies by SWEBOK area and activity.

SWEBOK area	**Activity**	**Article ID**
Requirements	Requirements modeling	[PS1]
	User story generation/enhancement	[PS2] [PS3]
Architecture and Design	Design modeling	[PS4] [PS5]
	UI/UX design	[PS6]
Construction	Code generation	[PS7] [PS10] [PS13] [PS16] [PS18] [PS20] [PS23] [PS24] [PS37] [PS39] [PS41] [PS42] [PS43] [PS44] [PS45]
	Code translation	[PS37] [PS39] [PS41] [PS43] [PS44] [PS45]
	Code completion/suggestion	[PS11] [PS19] [PS37] [PS39] [PS41] [PS42] [PS43] [PS44] [PS45] [PS47]
	Code refactoring/improvement	[PS8] [PS37] [PS38] [PS39] [PS41] [PS42] [PS43] [PS44] [PS45]
	Code documentation/comprehension	[PS9] [PS17] [PS21] [PS22] [PS37] [PS39] [PS40] [PS41] [PS42] [PS43] [PS44] [PS45]
	Version control and integration	[PS14] [PS15]
Testing	Bug/defect detection/classification	[PS25] [PS26] [PS30] [PS31] [PS32] [PS33] [PS34] [PS35] [PS36] [PS37] [PS38] [PS39] [PS41] [PS42] [PS43] [PS44] [PS45]
	Test case/data generation	[PS27] [PS28] [PS29] [PS37] [PS39] [PS41] [PS42] [PS43] [PS44] [PS45]
Maintenance	System tracking	[PS46] [PS47]
	Incident management	[PS48]

Table 6: Contribution of selected studies.

Article ID	Contribution facet	Contribution
[PS1]	Method/Approach	Provides a benchmark dataset for the automatic generation of domain models from natural language software requirements.
[PS2]	Model/Framework	Implements an autonomous LLM-based agent system (ALAS) at Austrian Post Group IT to automate the enhancement of user story quality in agile software development.
[PS3]	Method/Approach	Introduces an innovative AI-powered approach to transform unstructured software requirement texts into standardized user stories.
[PS4]	Tool	Presents a prototype of an AI-based chatbot designed to assist users of a MDE tool.
[PS5]	Model/Framework	Presents DesignSystemsJS, a JavaScript library that standardizes Design Systems to address inconsistencies and inefficiencies across teams.
[PS6]	Method/Approach	Proposes an approach to help domain experts create models in MDE by generating context-specific prompts that overcome limitations such as context size and attention fading.
[PS7]	Model/Framework	Integrates LLMs with Flow-Based Programming (FBP) to facilitate real-time code generation.
[PS8]	Tool	Describes the implementation of an AI-powered code review assistant to improve pull request management.
[PS9]	Method/Approach	Introduces DOSA, an approach that matches Stack Overflow (SO) questions to official documentation, aiming to bridge the gap between community-driven knowledge and formal resources.
[PS10]	Tool	Presents a system that automates the transformation of sketched wireframes into UI skeleton code, streamlining the design process and reducing manual effort.
[PS11]	Model/Framework	Introduces CCTEST, a framework designed to test and repair code completion systems in black-box settings.
[PS12]	Model/Framework	Presents CAPIR (Compositional API Recommendation), an approach to API recommendation for library-oriented code generation.
[PS13]	Method/Approach	Presents GAP-Gen, a Guided Automatic Python Code Generation method, designed to improve the efficiency of generating Python source code.

Continued on next page

Table 6 – continued from previous page

Article ID	Contribution facet	Contribution
[PS14]	Method/Approach	Introduces an approach, named CHATMERGE, for resolving Git merge conflicts in collaborative software development.
[PS15]	Method/Approach	Presents an approach, TDREMINDER, for detecting TODO-missed commits in software repositories, helping developers ensure long-term code quality and reduce technical debt.
[PS16]	Tool	Introduces KubePlaybook, a repository of Ansible playbooks designed for Kubernetes auto-remediation.
[PS17]	Tool	Discusses LARCH (Large Language Model-based Automatic Readme Creation with Heuristics), a system developed to automatically generate coherent and factually correct readmes for software repositories.
[PS18]	Model/Framework	Introduces LLMSecGuard, an open-source framework designed to enhance code security by integrating LLMs with static code analyzers.
[PS19]	Model/Framework	Introduces MicroRec, a novel microservice recommendation to improve the discovery, evaluation and compatibility of microservices.
[PS20]	Method/Approach	Presents an approach to translate human language requirements into high-level programming languages.
[PS21]	Method/Approach	Proposes P-CodeSum, an approach to Project-specific Code Summarization (PCS) that uses a neural prompt selector to generate high-quality, project-specific code summaries.
[PS22]	Method/Approach	Presents an approach for improving the performance of binary code comment quality classification models.
[PS23]	Model/Framework	Discusses UniLog, an automatic logging framework to generate log messages, determine logging positions and set verbosity levels.
[PS24]	Method/Approach	Proposes an approach called ChatDANCE to generate high quality and diverse augmented data to improve the performance of semantic code search models.
[PS25]	Method/Approach	Applies Latent Semantic Indexing (LSI) to cluster safety findings in industrial software development.
[PS26]	Method/Approach	Describes a method for automating bug triage using bug report content to assign developers and components.

Continued on next page

Table 6 – continued from previous page

Article ID	Contribution facet	Contribution
[PS27]	Method/Approach	Investigates the feasibility of using LLMs to automatically generate test cases from informal bug reports.
[PS28]	Model/Framework	Presents CAT-LM, a GPT-style language model with 2.7 billion parameters, trained on aligned code and tests to improve automated test generation.
[PS29]	Method/Approach	Introduces MuTAP, a method that enhances test case generation using mutation testing to better reveal bugs.
[PS30]	Method/Approach	Proposes an approach to improve the performance of Pre-trained Language Models of Code (PLMCs) for Automated Program Repair (APR) through source code augmentation and curriculum learning.
[PS31]	Model/Framework	Introduces CodeRanker, a neural ranker designed to predict the correctness of a sampled program without execution.
[PS32]	Tool	Introduces Fuzz4All, a universal fuzzing tool for generating diverse and realistic input for a wide range of programming and formal languages.
[PS33]	Tool	Proposes GAMMA, an approach for donor code generation in template-based APR.
[PS34]	Model/Framework	Introduces InferFix, an end-to-end program repair framework to fix critical security and performance bugs in software.
[PS35]	Tool	Presents MissConf, the first LLM-enhanced automated tool designed to reproduce Configuration-Triggered Bugs (CTBugs).
[PS36]	Model/Framework	Introduces MSDD-$(IA)^3$, a parameter-efficient multi-classification Software Defect Detection (SDD) framework.
[PS37]	Tool	Integrates a conversational AI virtual avatar to help users learn Pair Programming (PP) methodologies.
[PS38]	Method/Approach	Presents an AI-driven pipeline for detecting and refactoring data clumps in software development.
[PS40]	Method/Approach	Presents a web-based AI tool for rapid source code development using ChatGPT and prompt engineering.

Continued on next page

Table 6 – continued from previous page

Article ID	Contribution facet	Contribution
[PS41]	Model/Framework	Introduces CodeFuse-13B, an open-source, pre-trained multilingual LLM specifically designed for code-related tasks with English and Chinese prompts.
[PS42]	Tool	Introduces CARET, a conversational assistant for Java development in Eclipse to assist with software development tasks.
[PS43]	Model/Framework	Introduces F-CodeLLM, a federated learning framework designed to adapt LLMs for software development tasks while preserving code data privacy.
[PS44]	Tool	Introduces C4SE, a chatbot designed to assist software engineers and managers with various tasks throughout the software development lifecycle.
[PS45]	Tool	Discusses the development and evaluation of the Programmer's Assistant, a prototype system designed to explore the utility conversational interactions with a code-fluent LLM in software development.
[PS46]	Tool	Discusses the design and development of a log management platform that uses LLMs to improve log parsing and anomaly detection.
[PS47]	Model/Framework	Proposes LLMParser, a tool leveraging generative LLMs and few-shot tuning for log parsing, a critical first step in log-based analyses.
[PS48]	Method/Approach	Explores the use of X-lifecycle data for enhancing cloud incident management.

References

1. Adapa, C., Avulamanda, S.S., Anjana, A.R.K., Victor, A.: AI-Powered Code Review Assistant for Streamlining Pull Request Merging. In: 2024 IEEE International Conference for Women in Innovation, Technology & Entrepreneurship (ICWITE). pp. 323–327 (Feb 2024). `https://doi.org/10.1109/ICWITE59797.2024.10503540`
2. Adefris, B.B., Habtie, A.B., Taye, Y.G.: Automatic Code Generation From Low Fidelity Graphical User Interface Sketches Using Deep Learning. In: 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA). pp. 1–6 (Nov 2022). `https://doi.org/10.1109/ICT4DA56482.2022.9971204`
3. Alsayed, A.S., Dam, H.K., Nguyen, C.: MicroRec: Leveraging Large Language Models for Microservice Recommendation. In: Proceedings of the 21st International Conference on Mining Software Repositories. pp. 419–430. MSR '24, Association for Computing Machinery, New York, NY, USA (Jul 2024). `https://doi.org/10.1145/3643991.3644916`

4. Baumgartner, N., Iyenghar, P., Schoemaker, T., Pulvermüller, E.: AI-Driven Refactoring: A Pipeline for Identifying and Correcting Data Clumps in Git Repositories. Electronics **13**(9), 1644 (Jan 2024). https://doi.org/10.3390/electronics13091644, number: 9 Publisher: Multidisciplinary Digital Publishing Institute
5. Bourque, P., Fairley, R.E., Society, I.C.: Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0. IEEE Computer Society Press, Washington, DC, USA, 3rd edn. (2014)
6. Bozyigit, F., Bardakci, T., Khalilipour, A., Challenger, M., Ramackers, G., Babur, Ö., Chaudron, M.R.V.: Generating domain models from natural language text using NLP: a benchmark dataset and experimental comparison of tools. Software and Systems Modeling **23**(6), 1493–1511 (Dec 2024). https://doi.org/10.1007/s10270-024-01176-y
7. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language Models are Few-Shot Learners. In: Advances in Neural Information Processing Systems. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020)
8. Cai, Z., Chen, J., Chen, W., Wang, W., Zhu, X., Ouyang, A.: F-CodeLLM: A Federated Learning Framework for Adapting Large Language Models to Practical Software Development. In: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings. pp. 416–417. ICSE-Companion '24, Association for Computing Machinery, New York, NY, USA (May 2024). https://doi.org/10.1145/3639478.3643533
9. Contreras, A., Guerra, E., De Lara, J.: Conversational Assistants for Software Development: Integration, Traceability and Coordination:. In: Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering. pp. 27–38. SCITEPRESS - Science and Technology Publications, Angers, France (2024). https://doi.org/10.5220/0012561600003687
10. Dakhel, A.M., Nikanjam, A., Majdinasab, V., Khomh, F., Desmarais, M.C.: Effective test generation using pre-trained Large Language Models and mutation testing. Information and Software Technology **171**, 107468 (Jul 2024). https://doi.org/10.1016/j.infsof.2024.107468
11. De Vito, G., Lambiase, S., Palomba, F., Ferrucci, F.: Meet C4SE: Your New Collaborator for Software Engineering Tasks. In: 2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 235–238 (Sep 2023). https://doi.org/10.1109/SEAA60479.2023.00044, iSSN: 2376-9521
12. Dhyani, P., Nautiyal, S., Negi, A., Dhyani, S., Chaudhary, P.: Automated API Docs Generator using Generative AI. In: 2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS). pp. 1–6 (Feb 2024). https://doi.org/10.1109/SCEECS61402.2024.10482119, iSSN: 2688-0288
13. Di, P., Li, J., Yu, H., Jiang, W., Cai, W., Cao, Y., Chen, C., Chen, D., Chen, H., Chen, L., Fan, G., Gong, J., Gong, Z., Hu, W., Guo, T., Lei, Z., Li, T., Li, Z., Liang, M., Liao, C., Liu, B., Liu, J., Liu, Z., Lu, S., Shen, M., Wang, G., Wang, H., Wang, Z., Xu, Z., Yang, J., Ye, Q., Zhang, G., Zhang, Y., Zhao, Z., Zheng, X., Zhou, H., Zhu, L., Zhu, X.: CodeFuse-13B: A Pretrained Multi-lingual Code Large Language Model. In: Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice. pp. 418–429. ICSE-

SEIP '24, Association for Computing Machinery, New York, NY, USA (May 2024). https://doi.org/10.1145/3639477.3639719
14. Fu, Y., Wang, T., Li, S., Ding, J., Zhou, S., Jia, Z., Li, W., Jiang, Y., Liao, X.: MissConf: LLM-Enhanced Reproduction of Configuration-Triggered Bugs. In: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings. pp. 484–495. ICSE-Companion '24, Association for Computing Machinery, New York, NY, USA (May 2024). https://doi.org/10.1145/3639478.3647635
15. Goel, D., Husain, F., Singh, A., Ghosh, S., Parayil, A., Bansal, C., Zhang, X., Rajmohan, S.: X-Lifecycle Learning for Cloud Incident Management using LLMs. In: Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering. pp. 417–428. FSE 2024, Association for Computing Machinery, New York, NY, USA (Jul 2024). https://doi.org/10.1145/3663529.3663861
16. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Nets. In: Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc. (2014)
17. Hao, S., Shi, X., Liu, H.: Exploring the Potential of Pre-Trained Language Models of Code for Automated Program Repair. Electronics **13**(7), 1200 (Jan 2024). https://doi.org/10.3390/electronics13071200, number: 7 Publisher: Multidisciplinary Digital Publishing Institute
18. Inala, J.P., Wang, C., Yang, M., Codas, A., Encarnación, M., Lahiri, S., Musuvathi, M., Gao, J.: Fault-Aware Neural Code Rankers. Advances in Neural Information Processing Systems **35**, 13419–13432 (Dec 2022)
19. Iyenghar, P., Otte, F., Pulvermueller, E.: AI-guided Model-Driven Embedded Software Engineering:. In: Proceedings of the 10th International Conference on Model-Driven Engineering and Software Development. pp. 395–404. SCITEPRESS - Science and Technology Publications, Online Streaming, — Select a Country — (2022). https://doi.org/10.5220/0011006200003119
20. Jin, M., Shahriar, S., Tufano, M., Shi, X., Lu, S., Sundaresan, N., Svyatkovskiy, A.: InferFix: End-to-End Program Repair with LLMs. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 1646–1656. ESEC/FSE 2023, Association for Computing Machinery, New York, NY, USA (Nov 2023). https://doi.org/10.1145/3611643.3613892
21. Kavian, A., Pourhashem Kallehbasti, M.M., Kazemi, S., Firouzi, E., Ghafari, M.: LLM Security Guard for Code. In: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering. pp. 600–603. EASE '24, Association for Computing Machinery, New York, NY, USA (Jun 2024). https://doi.org/10.1145/3661167.3661263
22. Killivalavan, S., Thenmozhi, D.: Software Metadata Classification based on Generative Artificial Intelligence. In: Proceedings of the Forum for Information Retrieval Evaluation (FIRE). CEUR Workshop Proceedings (2023)
23. Koreeda, Y., Morishita, T., Imaichi, O., Sogawa, Y.: LARCH: Large Language Model-based Automatic Readme Creation with Heuristics. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. pp. 5066–5070. CIKM '23, Association for Computing Machinery, New York, NY, USA (Oct 2023). https://doi.org/10.1145/3583780.3614744
24. Kulkarni, V., Reddy, S., Barat, S., Dutta, J.: Toward a Symbiotic Approach Leveraging Generative AI for Model Driven Engineering. In: 2023 ACM/IEEE 26th

International Conference on Model Driven Engineering Languages and Systems (MODELS). pp. 184–193 (Oct 2023). https://doi.org/10.1109/MODELS58315.2023.00039

25. Kumar Dipongkor, A.: An Ensemble Method for Bug Triaging using Large Language Models. In: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings. pp. 438–440. ICSE-Companion '24, Association for Computing Machinery, New York, NY, USA (May 2024). https://doi.org/10.1145/3639478.3641228
26. Lee, E., Gong, J., Cao, Q.: Object Oriented BDD and Executable Human-Language Module Specification. In: 2023 26th ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter). pp. 127–133 (Jul 2023). https://doi.org/10.1109/SNPD-Winter57765.2023.10223873
27. Li, Y., Shi, J., Zhang, Z.: An Approach for Rapid Source Code Development Based on ChatGPT and Prompt Engineering. IEEE Access **12**, 53074–53087 (2024). https://doi.org/10.1109/ACCESS.2024.3385682, conference Name: IEEE Access
28. Li, Z., Wang, C., Liu, Z., Wang, H., Chen, D., Wang, S., Gao, C.: CCTEST: Testing and Repairing Code Completion Systems. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). pp. 1238–1250 (May 2023). https://doi.org/10.1109/ICSE48619.2023.00110, iSSN: 1558-1225
29. Ma, Z., An, S., Xie, B., Lin, Z.: Compositional API Recommendation for Library-Oriented Code Generation. In: Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension. pp. 87–98. ICPC '24, Association for Computing Machinery, New York, NY, USA (Jun 2024). https://doi.org/10.1145/3643916.3644403
30. Ma, Z., Chen, A.R., Kim, D.J., Chen, T.H.P., Wang, S.: LLMParser: An Exploratory Study on Using Large Language Models for Log Parsing. In: Proceedings of the 46th International Conference on Software Engineering (ICSE 2024). pp. 1209–1221. IEEE Computer Society (Apr 2024). https://doi.org/10.1145/3597503.3639150
31. Manfredi, G., Erra, U., Gilio, G.: A Mixed Reality Approach for Innovative Pair Programming Education with a Conversational AI Virtual Avatar. In: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering. pp. 450–454. EASE '23, Association for Computing Machinery, New York, NY, USA (Jun 2023). https://doi.org/10.1145/3593434.3593952
32. Namrud, Z., Sarda, K., Litoiu, M., Shwartz, L., Watts, I.: KubePlaybook: A Repository of Ansible Playbooks for Kubernetes Auto-Remediation with LLMs. In: Companion of the 15th ACM/SPEC International Conference on Performance Engineering. pp. 57–61. ICPE '24 Companion, Association for Computing Machinery, New York, NY, USA (May 2024). https://doi.org/10.1145/3629527.3653665
33. Oswal, J.U., Kanakia, H.T., Suktel, D.: Transforming Software Requirements into User Stories with GPT-3.5 -: An AI-Powered Approach. In: 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT). pp. 913–920 (Jan 2024). https://doi.org/10.1109/IDCIoT59759.2024.10467750
34. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering. pp. 68–77. EASE'08, BCS Learning & Development Ltd., Swindon, GBR (Jun 2008)

35. Plein, L., Ouédraogo, W.C., Klein, J., Bissyandé, T.F.: Automatic Generation of Test Cases based on Bug Reports: a Feasibility Study with Large Language Models. In: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings. pp. 360–361. ICSE-Companion '24, Association for Computing Machinery, New York, NY, USA (May 2024). https://doi.org/10.1145/3639478.3643119
36. Pudari, R., Zhou, S., Ahmed, I., Dai, Z., Zhou, S.: Aligning Documentation and Q&A Forum through Constrained Decoding with Weak Supervision. In: 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 346–351 (Oct 2023). https://doi.org/10.1109/ICSME58846.2023.00043, iSSN: 2576-3148
37. Pulido, C.: CarlosPulidoHernandez/Generative-AI-in-Software-Engineering-A-Systematic-Mapping-Study-based-on-SWEBOK-Areas
38. Rao, N., Jain, K., Alon, U., Goues, C.L., Hellendoorn, V.J.: CAT-LM Training Language Models on Aligned Code and Tests. In: Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering. pp. 409–420. ASE '23, IEEE Press, Echternach, Luxembourg (Sep 2024). https://doi.org/10.1109/ASE56229.2023.00193
39. Ross, S.I., Martinez, F., Houde, S., Muller, M., Weisz, J.D.: The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development. In: Proceedings of the 28th International Conference on Intelligent User Interfaces. pp. 491–514. IUI '23, Association for Computing Machinery, New York, NY, USA (Mar 2023). https://doi.org/10.1145/3581641.3584037
40. Shah, H., Kamuni, N.: DesignSystemsJS - Building a Design Systems API for aiding standardization and AI integration. In: 2023 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA). pp. 83–89 (Dec 2023). https://doi.org/10.1109/CoNTESA61248.2023.10384889
41. Shen, C., Yang, W., Pan, M., Zhou, Y.: Git Merge Conflict Resolution Leveraging Strategy Classification and LLM. In: 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS). pp. 228–239 (Oct 2023). https://doi.org/10.1109/QRS60937.2023.00031, iSSN: 2693-9177
42. van Solingen (Revision), R., Basili (Original article, 1994 ed.), V., Caldiera (Original article, 1994 ed.), G., Rombach (Original article, 1994 ed.), H.D.: Goal Question Metric (GQM) Approach. In: Encyclopedia of Software Engineering. John Wiley & Sons, Ltd (2002). https://doi.org/10.1002/0471028959.sof142
43. Stol, K.J., Fitzgerald, B.: Guidelines for Conducting Software Engineering Research. In: Felderer, M., Travassos, G.H. (eds.) Contemporary Empirical Methods in Software Engineering, pp. 27–62. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-32489-6_2
44. Sun, Y., Chen, Y., Zhao, H., Peng, S.: Design and Development of a Log Management System Based on Cloud Native Architecture. In: 2023 9th International Conference on Systems and Informatics (ICSAI). pp. 1–6 (Dec 2023). https://doi.org/10.1109/ICSAI61474.2023.10423328
45. Vidan, A., Fiedler, L.: A Composable Just-In-Time Programming Framework with LLMs and FBP. In: 2023 IEEE High Performance Extreme Computing Conference (HPEC). pp. 1–8 (Sep 2023). https://doi.org/10.1109/HPEC58863.2023.10363587, iSSN: 2643-1971
46. Voggenreiter, M., Schneider, P., Gulraiz, A.: Aggregating Industrial Security Findings with Semantic Similarity-Based Techniques. In: Abbas, M. (ed.) Practical

Solutions for Diverse Real-World NLP Applications, pp. 121–139. Springer International Publishing, Cham (2024). https://doi.org/10.1007/978-3-031-44260-5_7

47. Wang, H., Gao, Z., Hu, X., Lo, D., Grundy, J., Wang, X.: Just-In-Time TODO-Missed Commits Detection. IEEE Transactions on Software Engineering **50**(11), 2732–2752 (Nov 2024). https://doi.org/10.1109/TSE.2024.3405005, conference Name: IEEE Transactions on Software Engineering
48. Wang, X., Lu, L., Yang, Z., Tian, Q., Lin, H.: Parameter-Efficient Multi-classification Software Defect Detection Method Based on Pre-trained LLMs. International Journal of Computational Intelligence Systems **17**(1), 152 (Jun 2024). https://doi.org/10.1007/s44196-024-00551-3
49. Wang, Y., Guo, L., Shi, E., Chen, W., Chen, J., Zhong, W., Wang, M., Li, H., Zhang, H., Lyu, Z., Zheng, Z.: You Augment Me: Exploring ChatGPT-based Data Augmentation for Semantic Code Search. In: 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 14–25 (Oct 2023). https://doi.org/10.1109/ICSME58846.2023.00014, iSSN: 2576-3148
50. Xia, C.S., Paltenghi, M., Le Tian, J., Pradel, M., Zhang, L.: Fuzz4All: Universal Fuzzing with Large Language Models. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. pp. 1–13. ICSE '24, Association for Computing Machinery, New York, NY, USA (Apr 2024). https://doi.org/10.1145/3597503.3639121
51. Xu, J., Cui, Z., Zhao, Y., Zhang, X., He, S., He, P., Li, L., Kang, Y., Lin, Q., Dang, Y., Rajmohan, S., Zhang, D.: UniLog: Automatic Logging via LLM and In-Context Learning. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. pp. 1–12. ICSE '24, Association for Computing Machinery, New York, NY, USA (Feb 2024). https://doi.org/10.1145/3597503.3623326
52. Yun, S., Lin, S., Gu, X., Shen, B.: Project-specific code summarization with in-context learning. Journal of Systems and Software **216**, 112149 (Oct 2024). https://doi.org/10.1016/j.jss.2024.112149
53. Zhang, Q., Fang, C., Zhang, T., Yu, B., Sun, W., Chen, Z.: Gamma: Revisiting Template-based Automated Program Repair via Mask Prediction. In: Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering. pp. 535–547. ASE '23, IEEE Press, Echternach, Luxembourg (Sep 2024). https://doi.org/10.1109/ASE56229.2023.00063
54. Zhang, Z., Rayhan, M., Herda, T., Goisauf, M., Abrahamsson, P.: LLM-Based Agents for Automating the Enhancement of User Story Quality: An Early Report. In: Šmite, D., Guerra, E., Wang, X., Marchesi, M., Gregory, P. (eds.) Agile Processes in Software Engineering and Extreme Programming. pp. 117–126. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-61154-4_8
55. Zhao, J., Song, Y., Wang, J., Harris, I.: GAP-Gen: Guided Automatic Python Code Generation. In: Bassignana, E., Lindemann, M., Petit, A. (eds.) Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop. pp. 37–51. Association for Computational Linguistics, Dubrovnik, Croatia (May 2023). https://doi.org/10.18653/v1/2023.eacl-srw.4