Impact of Generative Artificial Intelligence on Knowledge Management in Software Engineering: A Systematic Mapping Study

Ana Claudia M. P. Costa
Julia Beiroco O. Fantini
Federal University of Technology – Paraná, Cornélio Procópio, Brazil
anaclaudiacosta@alunos.utfpr.edu.br
juliabeiroco@alunos.utfpr.edu.br

Érica Ferreira de Souza
Federal University of Technology – Paraná
Cornélio Procópio, Brazil
ericasouza@utfpr.edu.br

Glaucia Braga e Silva
Federal University of Viçosa (UFV)
Florestal, MG, Brazil
glaucia@ufv.br

Luciana Rebelo
Gran Sasso Science Institute (GSSI)
L'Aquila, Italy
luciana.rebelo@gssi.it

Katia Romero Felizardo
Federal University of Technology – Paraná, Cornélio Procópio, Brazil
katiascannavino@utfpr.edu.br

Giovani Volnei Meinerz
Federal University of Technology – Paraná, Cornélio Procópio, Brazil
giovanimeinerz@utfpr.edu.br

ABSTRACT

Context: In recent years, Generative Artificial Intelligence (GenAI) has emerged as a transformative technology with high potential across various organizational contexts. Leveraging techniques such as Natural Language Processing (NLP) and Large Language Models (LLMs), tools like ChatGPT, GitHub Copilot, and DALL-E have begun to facilitate information creation and retrieval, automate tasks, and optimize decision-making processes. In the field of Knowledge Management (KM), GenAI has served as an enabler for organizing, disseminating, and reusing knowledge, fostering more collaborative and responsive environments. Specifically in Software Engineering, GenAI demonstrates potential to enhance productivity. The integration of GenAI with KM practices can play a key role in improving software quality, by enabling better reuse of knowledge, supporting consistent practices, and promoting informed decision-making. Understanding how GenAI can support KM in this domain is therefore essential to guide its strategic and effective integration. **Objective**: This study aims to investigate the impact of GenAI on KM processes within organizations operating in the Software Engineering domain. **Method**: To achieve this, we conducted a systematic mapping study to identify current practices, perceptions, and challenges related to the use of GenAI in KM within Software Engineering. **Results:** The study analyzed 97 primary studies published between 2021 and 2025. Main results indicate that GenAI focusing on code generation and software construction activities. KM practices most commonly supported include knowledge application and creation, with externalization emerging as the dominant SECI knowledge conversion mode. **Conclusion:** GenAI contributes significantly to KM in Software Engineering by supporting operational tasks and the formalization of knowledge. However, limited attention to knowledge dissemination and deeper learning processes reveals promising directions for future research.

KEYWORDS

Software Engineering, Knowledge Management, Generative Artificial Intelligence, Systematic Mapping Study

1 Introduction

In recent years, there has been significant progress in Generative Artificial Intelligence (GenAI), a class of models capable of producing new content, such as images, text, code, and music, based on patterns learned from large datasets [1], [2]. Tools like ChatGPT, GitHub Copilot, and DALL-E exemplify this trend, highlighting the growing impact of such technologies on individuals and organizations alike [3]. These systems rely on Natural Language Processing (NLP) techniques and Large Language Models (LLMs), enabling the efficient creation and retrieval of information from massive repositories, with applications ranging from machine translation and text summarization to question answering [4], [5].

In the field of Knowledge Management (KM), GenAI has emerged as a facilitating technology, particularly in information-intensive tasks such as capturing, categorizing, and disseminating both tacit and explicit knowledge [6], [7]. From a KM perspective, the capabilities of GenAI, such as natural language understanding, contextual reasoning, and content generation, align closely with core processes like knowledge creation, storage, transfer, and application. For instance, GenAI tools can support the externalization of tacit knowledge by transforming experiential insights into structured documentation, or enhance the combination of explicit knowledge by synthesizing data from multiple sources into coherent narratives.

Models like ChatGPT, based on the Generative Pretrained Transformer (GPT) architecture, are reshaping organizational processes, from document generation to intelligent user interactions through chatbots and virtual assistants [5]. Beyond automating cognitive tasks, these systems demonstrate continuous adaptability and context-awareness, making them strategic allies for KM in dynamic and knowledge-intensive environments. Recent studies suggest that GenAI accelerates knowledge flows, reduces informational silos, and supports evidence-based decision-making [6], [3]. Consequently, GenAI is increasingly viewed not just as a tool that supports KM, but as a transformative force capable of reshaping how organizations create, share, and utilize knowledge.

In the context of Software Engineering, the use of GenAI has gained growing interest due to its potential to boost productivity

and reduce cognitive effort in tasks such as coding, testing, and documentation [8], [9]. Empirical studies report productivity gains ranging from 20% to 45% with the use of LLM-based assistants such as GitHub Copilot, especially in repetitive or low-complexity programming tasks [10], [11]. However, despite these promising benefits, its practical adoption still faces several challenges, including technical (e.g., output accuracy and control), ethical (e.g., bias and transparency), and organizational (e.g., training and cultural adaptation) aspects [12], [13].

Leveraging KM practices, GenAI can enhance software quality by improving knowledge reuse, fostering standardization, and supporting more informed decisions. Understanding how GenAI supports KM in Software Engineering is essential to guide its adoption in ways that strategically improve software quality and development effectiveness. This paper investigates the impacts of GenAI on KM processes within organizations in the Software Engineering domain. Specifically, it examines how these technologies act as catalysts that accelerate and enable knowledge flows by transforming practices related to knowledge creation, organization, use, and dissemination. To achieve this, a systematic mapping study was conducted to identify and synthesize existing academic research on the role of GenAI in Software Engineering, with emphasis on KM.

The remainder of this paper is organized as follows. Section 2 introduces the background and key concepts. Section 3 describes the research method adopted for the mapping. Section 4 details the study selection process. The results, guided by the research questions, are presented in Section 5. Section 6 discusses the findings. Related work is highlighted in Section 7, and Section 8 presents the conclusions and directions for future research.

2 Background

In this section, the main concepts of this study and related work are discussed.

2.1 Generative Artificial Intelligence (GenAI)

Generative Artificial Intelligence (GenAI) refers to AI techniques focused on producing original outputs by learning complex patterns from extensive datasets. Unlike traditional AI systems that operate primarily through classification or rule-based decision-making, GenAI models are designed to generate content that resembles human-created artifacts. This includes text, visual art, and source code. The emergence of widely accessible tools, such as ChatGPT for conversational generation, GitHub Copilot for code assistance, and DALL-E for image creation, has made GenAI a prominent force in both consumer applications and professional domains [1], [3].

GenAI systems leverage various underlying technologies to enable content generation. Among the most influential are LLMs, trained on vast text corpora to perform natural-language tasks with remarkable fluency and coherence [14]. The GPT architecture, in particular, has become a cornerstone for many applications [15]. For visual content, Generative Adversarial Networks (GANs) play a central role, employing dual neural networks in a competitive setup to produce increasingly realistic outputs [16], [17]. Together, these models enable semantic understanding, context retention, and creative synthesis. By automating cognitive work, supporting rapid prototyping, and enhancing user interaction, GenAI fosters innovation and productivity across domains [1], [2].

2.2 Knowledge Management (KM)

Knowledge Management (KM) refers to the systematic process of creating, organizing, sharing, and applying knowledge within organizations. Its goal is to make organizational knowledge accessible, reusable, and actionable [18]. A central conceptual foundation in KM is the distinction between tacit and explicit knowledge [7]. Tacit knowledge is personal, context-specific, and difficult to formalize. It includes beliefs, values, experiences, and intuition that are hard to express or codify. In contrast, explicit knowledge is objective and can be documented, stored, and transferred through manuals, databases, or written instructions.

To explain how knowledge flows through organizations, Nonaka and Takeuchi [7] proposed the **SECI model**, which describes the dynamic conversion between tacit and explicit forms. It comprises four modes: (i) **Socialization** (tacit-to-tacit), when knowledge is shared through experience; (ii) **Externalization** (tacit-to-explicit), when tacit insights are articulated into concepts, often using metaphors or models; (iii) **Combination** (explicit-to-explicit), when explicit knowledge sources are integrated into new structures; and (iv) **Internalization** (explicit-to-tacit), when explicit knowledge becomes internalized through learning or practice. The model emphasizes that organizational knowledge evolves through interaction and collaboration, essential in dynamic and innovative environments.

To operationalize KM, several **KM cycles** have been proposed to guide activities such as identifying, capturing, disseminating, and applying knowledge [19]. Among the best known are the models by Wiig [20], Meyer and Zack [21], McElroy [22], and Bukowitz and Williams [23]. Synthesizing elements of these, Dalkir [19] proposed an **integrated KM cycle** structured in three stages: (i) **knowledge capture and/or creation**, in which tacit knowledge is elicited and explicit knowledge codified; (ii) **knowledge sharing**, involving dissemination through meetings, documentation, or intranet platforms; and (iii) **knowledge acquisition and application**, when knowledge is applied to decision-making and performance improvement. These integrated cycle activities are frequently adopted as theoretical foundations in research involving KM practices [24, 25].

2.3 GenAI and KM

The popularization of GenAI, driven by the launch of ChatGPT, marked a significant shift in how knowledge is accessed and managed within organizations. Tools based on natural language models, such as ChatGPT, enable intelligent conversational interactions that support efficient information retrieval, synthesis, and dissemination [26]. These systems accelerate knowledge sharing and continuously adapt to users' contexts, becoming strategic allies in KM and enablers of new knowledge creation through novel content, insights, and solutions. Beyond chatbots, other GenAI applications also contribute directly to KM: GitHub Copilot assists developers by suggesting context-aware code snippets, while Adobe Sensei automates creative tasks, enhancing knowledge generation and reuse across domains [27]. By integrating generation, adaptation, and recommendation capabilities, GenAI is redefining how knowledge is captured, organized, and applied in modern organizations.

2.4 GenAI, KM and Software Engineering

In the Software Engineering context, software development companies engage in activities that generate a considerable amount of knowledge [24], [28]. KM principles have received increasing attention in this environment, as they enable both tacit and explicit knowledge generated in software development to be captured, shared, and applied to enhance organizational learning [28]. In traditional software development models, explicit knowledge tends to dominate, with a strong reliance on formal documentation throughout the lifecycle [29]. In contrast, Agile Software Development (ASD) emphasizes collaboration and favors tacit knowledge, which is exchanged through direct interactions and shared experiences. However, transforming this tacit knowledge into explicit artifacts remains a complex task, particularly in dynamic and fast-paced development environments [7], [24].

One of the persistent challenges in Software Engineering is the effective conversion and dissemination of knowledge within dynamic, team-based and often distributed environments [30], [31], [32]. This is where GenAI offers promising opportunities. By enabling the generation of context-aware content, such as summaries, documentation, test cases and even code snippets, GenAI tools can assist in externalizing tacit knowledge, making it explicit and accessible. In addition, their adaptive and interactive nature supports real-time knowledge sharing and contextual guidance, which are essential in fast-paced development settings. In this context, GenAI emerges not only as a technological enabler but also as a potential catalyst for KM in software organizations, helping bridge the gap between tacit insights and explicit artifacts and enhancing learning and decision-making across software projects.

3 Research Method

The systematic mapping presented in this work follows the methodological guidelines outlined by [33] and [34]. The process comprises three main phases: (i) **Planning**, which defines the review objective and establishes a protocol specifying research questions, selection criteria, databases, search string, and quality assessment procedures; (ii) **Execution**, where primary studies are selected and relevant data extracted and synthesized; and (iii) **Reporting**, focused on addressing the research questions and presenting key findings.

In the following, we describe the main protocol steps adopted in the execution of this mapping study.

Research Questions (RQs). Table 1 presents the RQs as well as the rationale for considering them.

Search String. The search string considers two areas - GenAI and SE. The areas and the search string adopted in this systematic mapping can be seen in Table 2. We executed the search string in three metadata fields: title, abstract and keywords.

Sources. We selected the Scopus database for this study, as it is recognized as the largest repository of abstracts and citations for peer-reviewed literature. Moreover, Scopus is widely used in the field of Computer Science [35, 36], comprising over 60 million indexed records. It is worth noting that Scopus includes publications from several major international publishers, such as Cambridge University Press, Association for Computing Machinery (ACM), Institute of Electrical and Electronics Engineers (IEEE), Nature Publishing Group, Springer, Wiley-Blackwell, and Elsevier.

Selection Criteria. The selection criteria are organized in one inclusion criteria (IC) and eight exclusion criteria (EC). The inclusion critera is **(IC1)** The study should address Generative Artificial Intelligence in the context of Software Engineering. The exclusion criteria are: **(EC1)** Abstracts or extended abstracts without the corresponding full text; **(EC2)** The study is not written in English; **(EC3)** Duplicated works or earlier versions of publications already included; **(EC4)** Non-primary studies, including editorials, keynote summaries, tutorials, posters, conference reports, secondary studies such as systematic reviews or mappings, and literature surveys; **(EC5)** Full-text not available; **(EC6)** The study does not address Generative Artificial Intelligence within the context of Software Engineering; and **(EC7)** Although the study qualifies as a primary study, its focus is restricted to tool description or comparison, without demonstrating practical application in SE.

Data Storage. The publications retrieved during the search phase were properly cataloged and stored. To support this process, a data extraction spreadsheet was created to record all relevant information from the selected studies. Each study received a unique identifier, facilitating tracking throughout the different selection stages. The catalog includes details about each selection phase, along with the necessary data to address the research questions and ensure effective management of the systematic mapping process. This structured catalog was instrumental in guiding the classification and analysis steps.

Assessment. Prior to executing the mapping study, the proposed protocol was validated through a pilot test. This test aimed to assess its applicability and consistency using a set of studies previously identified as relevant to the research, referred to as the control group. Two studies were selected to compose this control group, namely [37], [38]. To construct the search string, we adopted an iterative approach: starting with an initial list of terms, we refined and adjusted them until all control group studies were successfully retrieved. During the pilot tests, when the term "Knowledge Management" was included in the search string, we observed that very few studies were returned. This occurred because most papers do not explicitly address Knowledge Management; instead, they discuss it implicitly through related practices such as knowledge sharing, documentation, or learning processes. Therefore, we decided not to include the term "Knowledge Management" directly in the final search string to avoid narrowing the results and missing relevant studies.

The mapping activities were carried out collaboratively by all authors. The first author coordinated and actively participated in all stages of the study, leading the planning, execution, and reporting phases. However, all authors were involved in key activities such as study selection, data extraction, and synthesis. Decisions and divergences identified during the process were validated by the authors with the highest expertise in the topic to ensure methodological rigor and consistency across the review. The initial pool of studies was evenly distributed among the team members. Throughout the selection, extraction, and synthesis phases, we periodically exchanged papers among authors to incorporate different perspectives and minimize individual biases. In addition, we held regular

Table 1: Research questions and their rationales

Nº	Research Question	Rationale
RQ1	What is the main objective or motivation behind the reported studies involving GenAI in the context of Software Engineering and Knowledge Management?	Understanding study motivations helps identify the main research drivers and practical needs that GenAI addresses in Software Engineering and KM, revealing its strategic intent toward productivity, knowledge dissemination, innovation, or process improvement.
RQ2	What types of GenAI approaches or tools are adopted in the identified studies?	Mapping GenAI approaches and tools provides a technical overview of the field, identifying which models, frameworks, or systems (e.g., ChatGPT, Copilot) are most used and for what purposes, supporting reproducibility and technological relevance.
RQ3	Which Software Engineering activities are addressed through the use of GenAI	Identifying targeted Software Engineering activities (e.g., coding, testing, documentation) shows where GenAI is applied and in which development phases it adds most value, highlighting opportunities and gaps for future integration.
RQ4	Which Knowledge Management activities are considered in the context of GenAI applications?	Determines how GenAI supports KM activities and whether it contributes more to knowledge externalization, transfer, or reuse.
RQ5	What impacts of GenAI are reported regarding knowledge conversion processes (e.g., tacit to explicit knowledge)?	As knowledge conversion is central in KM (SECI model), this question explores whether GenAI facilitates the articulation and formalization of tacit knowledge, a major challenge in Software Engineering.

Table 2: Systematic mapping - Areas and Keywords

Areas	Keywords
Generative Artificial Intelligence	*"Chat GPT" OR "ChatGPT" OR "chatbot" OR "copilot" OR "generative AI" OR "generative artificial intelligence"*
Software Engineering	*"software engineering" OR "computer science" OR "information technology" OR "software develop*"*
Search String: *("Chat GPT" OR "ChatGPT" OR "chatbot" OR "copilot" OR "generative AI" OR "generative artificial intelligence") AND ("software engineering" OR "computer science" OR "information technology" OR "software develop*")*	

weekly meetings, inspired by the Scrum framework, where each author shared what had been done, identified any encountered issues, and outlined the next planned tasks. This practice contributed to maintaining alignment, transparency, and agility throughout the process.

4 Selection Process

Figure 1 illustrates the main stages followed in this SLR. The search string, detailed in Table 2, was applied to the Scopus database, targeting the title, abstract, and keywords fields. We considered publications from January 2014 to January 2025. This search returned a total of 1264 studies from the selected data source.

In the first stage, 13 duplicate articles were removed, resulting in 1251 studies. In the second stage, we applied the selection criteria (i.e., inclusion and exclusion criteria) to the titles, abstracts, and keywords, which reduced the set to 418 papers (a reduction of approximately 66.6%). In the third stage, the selection criteria were applied to the full texts, yielding 97 studies (a reduction of approximately 76.8%). As a final outcome, 97 studies were selected for analysis. The replication package, including the full list of references for the 97 selected studies, is available in the supplementary material to ensure reproducibility. The access link is provided at the end of this paper.

5 Data extraction and Synthesis

A total of 97 primary studies were selected through the SLR. Figure 2 illustrates the distribution of these studies by publication year. Although the search covered the past eleven years, relevant publications were identified only from 2021 onward. A clear upward trend is evident: 1 study (1%) in 2021, 4 studies (4.1%) in 2022, followed by a sharp increase in 2023 with 20 studies (20.6%). Notably, 2024 marks the peak of research activity so far, with 68 studies published (70.1%), reflecting the recent and accelerated growth of interest in this field. As of the data collection period, 4 studies (4.1%) had been published in 2025. Since the search was conducted in January 2025, this number reflects only the very beginning of the year, and we believe that the total number of publications in 2025 may eventually surpass that of 2024.

In terms of geographic distribution, authors from 34 different countries contributed to the selected studies, demonstrating the global reach of the topic. However, the contributions are highly concentrated: 5 of these 34 countries, representing 57.9% of the authors, correspond to studies involving authors from just five nations: The United States, China, Canada, Germany, and Italy. This indicates that, while the research is internationally distributed, scientific output in this area is primarily concentrated in a few countries with higher publication activity.

This recent surge in publications can be partially attributed to the release of OpenAI's ChatGPT in late 2022 and its subsequent technological advancements [39]. As further discussed in the response to Research Question 3 (RQ3), the majority of the analyzed studies focus on tools and solutions developed by OpenAI, reinforcing the correlation between the emergence of these technologies and the rapid growth of academic interest in the topic.

The following sections present the answers to each Research Question (RQs), along with analyses and interpretations derived

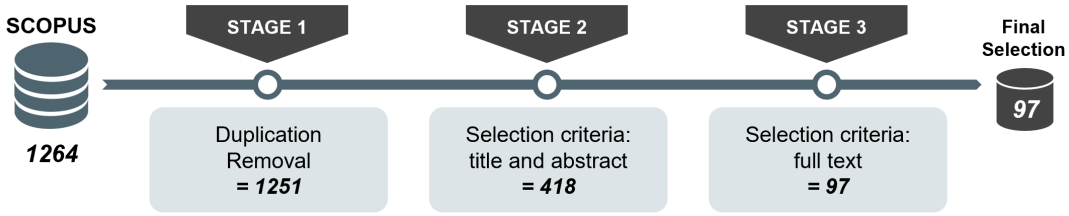

Figure 1: Search and selection SLR process

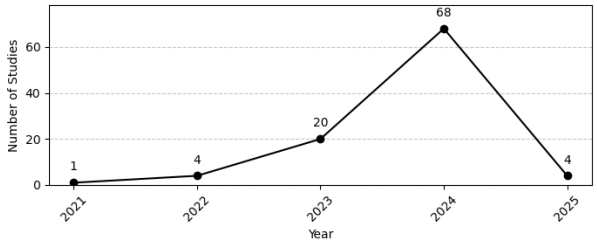

Figure 2: Number of publications per year.

from the selected studies. To support the discussion of key findings, the studies are referenced using both their formal citations and identifiers (IDs), as listed in the conduction spreadsheet and in the reference list available in the replication package. The **supplementary material** at the end of this article includes the conduction spreadsheet and the complete list of studies selected for analysis.

RQ1. *What is the main objective or motivation behind the reported studies involving GenAI in the context of Software Engineering and Knowledge Management?*

The objective of RQ1 was to identify and analyze the practical focus of the selected studies in terms of how GenAI has been applied within Software Engineering activities. To address RQ1, the purposes of each selected study were identified using a qualitative coding technique. This process consisted of systematically categorizing semantic patterns in the textual descriptions of the studies, allowing for the organization and interpretation of the practical objectives addressed [40]. As a result, two complementary categorization axes were defined: Proposal, which classifies the type of solution proposed or applied, and GenAI Usage, which captures the technical purpose of the GenAI implementation. The Proposal categorization was structured as follows:

- **New tool:** When the study proposed or developed a new Generative AI-based tool.
- **New framework:** When a methodology, conceptual framework, or structured process incorporating Generative AI as a core component was proposed.
- **Tool use**: When the study focused on the application of GenAI, either through the direct use of existing tools, systems, or processes.

In parallel, the GenAI Usage categorization identifies the primary technical application addressed in each study:

- **Code generation:** When GenAI was used to generate source code or scripts.
- **Task automation**: When GenAI supported task automation activities, without directly generating code.
- **Automated test generation**: when GenAI was used to create automated tests.
- **Artifact generation**: When GenAI was applied to generate formal Software Engineering artifacts, such as requirements, specifications, or technical documentation.

The results of this classification are summarized in Figure 3, which presents a heatmap combining both categorization axes. This visualization highlights the main research trends by illustrating the relationship between the type of proposal and the corresponding technical application of GenAI.

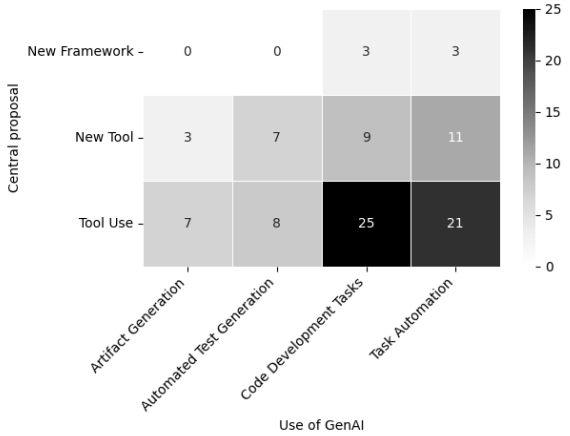

Figure 3: Relationship between proposal type and GenAI application focus

Analysis of the results reveals a clear predominance of studies classified under "**Tool Use**". Specifically, 65% of the analyzed studies focused on the application of GenAI, either through existing tools (such as ChatGPT and GitHub Copilot) or via the integration of APIs and LLMs into existing environments. Within this category, "**Code Generation**" emerged as the most common technical focus, appearing in approximately 58% of the studies, which reflects the central role of GenAI in supporting software development activities through source code generation.

In contrast, only **22%** of studies were categorized as "**New Tool**", representing efforts to develop dedicated solutions, such as the

Cypress Copilot [41] (ID 1), a tool designed to automate test generation within the Cypress framework. "**New framework**" studies were less frequent, highlighting that most methodological innovations remain secondary compared to tool development and direct application.

Overall, these results suggest that current research in this field prioritizes the practical adoption of GenAI through existing technologies rather than focusing primarily on the creation of new tools or conceptual frameworks.

RQ2. *What types of GenAI approaches or tools are adopted in the identified studies?*

The objective of RQ2 was to identify and classify the GenAI technologies employed in the analyzed studies, with the goal of mapping which tools, APIs, and foundational models are currently adopted in Software Engineering activities. This mapping aimed to provide a structured overview of the technological ecosystem that supports the use of GenAI in this domain.

To address RQ2, each study was analyzed to extract the specific GenAI technologies mentioned. These technologies were then categorized into three distinct groups:

- **Commercial Tools:** Solutions directly accessible to end-users, such as conversational agents or code assistants.
- **Tool APIs:** Interfaces providing programmatic access to GenAI functionalities.
- **Base Models (LLMs):** Foundational models serving as the technological core for various tools and systems.

This categorization aimed to differentiate between technologies used interactively by professionals, solutions embedded via API integrations, and models that operate as the foundation of both tools and APIs. The summarized results are presented in Table 3.

Table 3: Distribution of Generative AI Technologies (RQ2)

Category	Technology	Studies
Tool APIs	ChatGPT API	13
	Gemini API	2
Commercial Tools	ChatGPT	45
	Github Copilot	15
	TabNine	1
	Bing Chat	1
	Amazon Q Code Transformation	1
	Claude-3.5-Sonnet	1
Base Models (LLMs)	GPT-4	17
	GPT-3.5	14
	OpenAI Codex	7
	LLaMA-2	2
	GPT-3	1
	GPT-Neo	1
	LLaMA	1
	Code LLaMA	1
	PaLM	1
	Qwen-72B	1
	Themisto	1
	Ollama	1

Commercial tools represented the largest group, accounting for approximately 47.4% of all technology mentions. Within this category, ChatGPT was the most cited, appearing in around 33.3% of the studies, confirming its position as the primary tool adopted in practical applications. GitHub Copilot was identified in approximately 11.1% of the studies, reinforcing its relevance in code generation tasks, a finding consistent with the focus identified in RQ1.

Tool APIs represented about 11.1% of total mentions. In this category, it was observed that APIs, such as the ChatGPT API, were often referenced without detailing the underlying base model. This suggests that, in many cases, APIs are viewed primarily as integration components, with less emphasis placed on the underlying architecture or model specifics.

Base models (LLMs) accounted for approximately 34.1% of the mentions. Among these, GPT-4 was the most cited base model, appearing in about 11.8% of the studies, followed by GPT-3.5 (9.6%) and OpenAI Codex (5.2%). The presence of open-weight models like LLaMA-2 and Qwen-72B, although less frequent, points to an emerging trend towards the adoption of open alternatives, often motivated by factors such as licensing flexibility or specific research objectives.

In total, 135 technology mentions were identified across the studies, indicating that many studies employed more than one tool, API, or model, thus reflecting the presence of combined usage scenarios. The complete distribution of the technologies identified is presented in Table 3, structured according to the three defined categories. These results reinforce the predominance of OpenAI solutions in both tool and model adoption, while also revealing emerging diversification in technological choices within the Software Engineering context.

RQ3. *Which Software Engineering activities are addressed through the use of GenAI?*

The objective of RQ3 was to identify which Software Engineering activities are most represented in the practical applications of GenAI within the analyzed studies. To structure this classification, the official Knowledge Areas (KAs) defined by the SWEBOK Guide 4.0 [42] were adopted, providing a standardized framework that encompasses essential domains in Software Engineering, including Software Requirements, Software Construction, Software Testing, and Software Maintenance.

The classification was performed through a detailed content analysis of each selected study, mapping the primary activities supported by GeneAI to one or more SWEBOK KAs [42]. This process considered both explicit mentions (e.g., automated test generation classified under Software Testing) and inferred contributions (e.g., using ChatGPT to generate API documentation categorized as Software Documentation).

The analytical results, presented in Figure 4, indicate a strong concentration of applications within the Software Construction knowledge area, which was identified in 43 studies, the most frequently represented domain. This trend is exemplified by study [43] (ID 27), which focused on Java source code generation; study [44] (ID 57), which addressed PHP; and study [45] (ID 80), which explored code generation for Arduino platforms. Beyond code generation, GenAI has also been the subject of studies exploring its

role as a direct assistant to software engineers during development activities. Specifically, studies [46] (ID 93) and [47] (ID 95) evaluated GitHub Copilot as an automated partner in pair programming, assessing its effectiveness in replicating the supportive role typically performed by a human collaborator, and highlighting its potential to transform collaborative development practices.

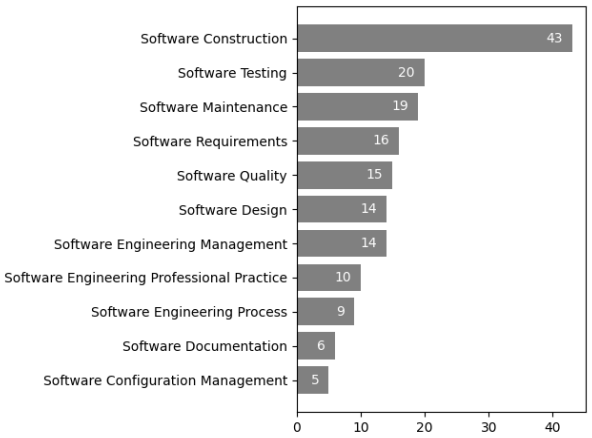

Figure 4: Distribution of activities by SWEBOK 4.0 Knowledge Areas

This result reinforces the trends observed in RQ1 and RQ2, which showed that the majority of studies focused on code generation using tools like ChatGPT and GitHub Copilot. These findings confirm that GenAI is primarily being leveraged to support operational activities that are directly related to software development.

Following Software Construction, considerable attention was also observed in Software Testing and Software Maintenance, both with approximately 20 occurrences. GenAI was applied to tasks such as automated test case generation, test data generation, bug fixing, and code refactoring, which emphasizes its relevance in the validation and maintenance phases of the software lifecycle.

Other knowledge areas, including Software Requirements and Software Quality, appeared with moderate frequency and included applications such as requirement elicitation, specification, functional size estimation, vulnerability detection, and assessment of code quality.

In contrast, areas such as Software Design, Software Engineering Management, and Software Engineering Process were less represented, although some relevant use cases were found, including the generation of domain models, project planning support, and automation of process documentation. Software Security, Software Configuration Management, and Software Documentation appeared only occasionally, typically associated with broader activities such as quality assurance or collaborative development.

Overall, the analysis indicates that current research on GenAI in Software Engineering is predominantly focused on construction, testing, and maintenance tasks. This reflects a practical emphasis on code-related activities, while project management, architectural, and foundational areas remain underexplored. These underrepresented areas offer promising directions for future investigations.

RQ4. *Which Knowledge Management activities are considered in the context of GenAI applications?*

The objective of RQ4 was to identify which stages of the integrated KM cycle, as proposed by Dalkir [48], are addressed in the analyzed studies. As presented in Section 2.2, this cycle consists of three core stages: knowledge capture and/or creation, knowledge sharing and dissemination, and knowledge acquisition and application. The first stage refers to the generation or formalization of knowledge, whether tacit or explicit. The second involves the distribution and communication of knowledge across the organization, while the third relates to the practical use of knowledge to solve problems and support decision-making processes.

To address this question, each of the 97 selected studies was individually analyzed and classified according to the KM activities explicitly reported, such as source code generation, documentation creation, resource dissemination, or the application of knowledge in real-world scenarios. A conservative classification approach was adopted to ensure that only clearly stated activities, directly aligned with Dalkir's KM cycle, were considered. The results, summarized in Figure 5, indicate that knowledge acquisition and application was the most frequently addressed stage, followed by knowledge capture and/or creation, and lastly, knowledge sharing and dissemination.

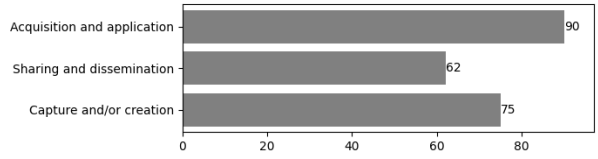

Figure 5: Distribution of activities by integrated KM cycle (Dalkir model)

The results indicate that knowledge acquisition and application is the most commonly addressed stage, explicitly present in 90 studies, representing 93.8% of the analyzed sample. This reflects a predominant focus on the operational use of AI-generated outputs, whether through automated code execution, deployment of generated artifacts, or application of produced documentation. Knowledge capture and/or creation was identified in 75 studies, equivalent to 77.3% of the total, indicating frequent use of Generative AI to create source code, models, or technical documentation. In contrast, knowledge sharing and dissemination was the least addressed stage, present in 62 studies, corresponding to 63.9%. This suggests that structured dissemination and reuse of AI-generated knowledge, beyond localized project contexts, remain secondary considerations in the current body of research.

These findings suggest that current applications of GenAI in Software Engineering primarily emphasize the generation and practical application of knowledge artifacts, with less attention given to systematic dissemination processes. This highlights a potential research opportunity to investigate how AI-generated knowledge can be more effectively shared and institutionalized across teams and organizations.

RQ5. *What impacts of GenAI are reported regarding knowledge conversion processes (e.g., tacit to explicit knowledge)?*

To analyze how Generative AI contributes to knowledge conversion in Software Engineering, this study adopts the SECI model proposed by Nonaka and Takeuchi [7], as outlined in Section 2.2. The model defines four modes of knowledge transformation: Socialization (tacit-to-tacit), Externalization (tacit-to-explicit), Combination (explicit-to-explicit), and Internalization (explicit-to-tacit). Each selected study was classified according to the specific conversion process explicitly supported or automated by GenAI.

Figure 6 presents the distribution of studies across the SECI quadrants. The numbers indicate how many studies were associated with each conversion mode. Notably, Figure 6 also includes circles that highlight studies mapped to more than one mode of knowledge conversion, these represent cases where GenAI supports multiple types of knowledge transformation within a single application or context. This visual cue emphasizes the multifaceted nature of GenAI's contribution in some studies.

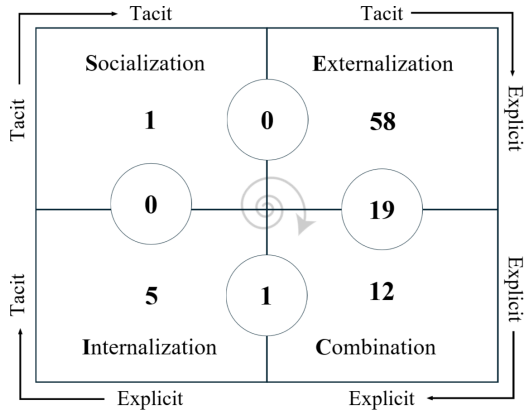

Figure 6: Distribution of activities by SECI knowledge conversion modes.

Socialization was the first mode represented, identified in just one study. In [49] (ID 45), team discussions around GenAI-generated outputs facilitate tacit-to-tacit knowledge exchange, illustrating how shared reflection can support informal knowledge transfer. The Externalization quadrant dominates the distribution, with 58 studies showing how GenAI converts tacit knowledge, such as natural language inputs or developer intuition, into explicit artifacts like source code, documentation, or formal models. This is evident in studies focused on documentation generation, such as [50] (ID 43) and [51] (ID 97), as well as in [52] (ID 29), which explores the transformation of UML diagrams into textual specifications using LLMs. Combination was identified in 12 studies, generally involving the integration of explicit knowledge sources. Examples include [53] (ID 18), which merges databases and knowledge graphs into unified repositories, and [54] (ID 28), which combines technical documentation with conversation logs to support CI/CD processes. Internalization appeared in 5 studies, where explicit outputs from GenAI tools are assimilated through practice. Study [55] (ID 41) uses eye-tracking in IDEs to observe how developers learn from and internalize LLM-generated code.

Additionally, 19 studies displayed hybrid characteristics across SECI modes, primarily between Externalization and Combination. For example, [56] (ID 14) introduces a tool that converts tacit prompts into explicit test datasets and integrates them with static data and APIs for testing. Similarly, [57] (ID 9) presents the Calico system, which performs Combination by embedding explicit annotations into code, subsequently internalized by the LLM and applied tacitly.

In summary, these findings indicate that GenAI is predominantly positioned as a catalyst for Externalization in Software Engineering, facilitating the conversion of professional experience and intuition into structured, explicit artifacts. The presence of multi-quadrant studies suggests that more complex or mature GenAI applications often span multiple knowledge conversion processes. The relative scarcity of studies in Socialization and Internalization also reveals a gap in current research regarding collaborative knowledge building and experiential learning-highlighting promising directions for future work.

6 Discussion

This study aimed to examine how GenAI has been applied in Software Engineering and how such applications intersect with KM processes. Based on the analysis of 97 primary studies published between 2021 and early 2025, the findings highlight an accelerated and globally distributed research movement that aligns with the emergence and rapid diffusion of LLM-based technologies such as ChatGPT and GitHub Copilot.

Overview and Temporal Patterns

Although the review covered publications from 2014 to 2025, relevant studies only emerged from 2021 onwards, confirming that GenAI represents a recent and evolving research frontier in SE. The exponential growth observed in 2023 and especially in 2024 demonstrates that the field's maturation coincides with the public release of accessible LLMs, particularly those from OpenAI. This pattern indicates a reactive and technology-driven research dynamic, in which academic exploration tends to follow the availability and diffusion of commercial tools rather than anticipate conceptual or methodological challenges.

From a technical perspective, the synthesis shows that the majority of studies (65%) emphasize tool usage over the development of new frameworks or original tools. This operational emphasis suggests that professionals are primarily exploring how GenAI can enhance existing SE workflows rather than rethinking or redesigning them. The prevalence of code generation (58%) as the dominant application aligns with SE's inherently practical orientation but also reveals an imbalance: its use remains heavily concentrated on automating programming tasks, while strategic, managerial, and architectural domains remain underexplored.

This trend reinforces a shift from innovation to adoption, in which short-term productivity gains often outweigh deeper investigations into long-term impacts, integration challenges, and ethical or organizational implications. Furthermore, few studies

advance theoretical models or methodological guidelines to systematize GenAI adoption, revealing a gap between practice-oriented experimentation and conceptual consolidation.

The technological landscape identified in RQ2 shows a strong concentration around OpenAI solutions, particularly ChatGPT, GPT-4, GPT-3.5, and Codex. While this dominance reflects the maturity and accessibility of these platforms, it also exposes potential risks of dependency on proprietary ecosystems that limit transparency, reproducibility, and academic neutrality. The modest but growing use of open-weight alternatives such as LLaMA-2 and Qwen-72B suggests an emerging countertrend toward decentralized and open research, which deserves further attention in future studies.

Knowledge Management Perspective

From a KM standpoint, the results indicate that GenAI applications mainly support knowledge capture and creation (77.3%) and knowledge acquisition and application (93.8%), while knowledge sharing and dissemination (63.9%) remain secondary. This imbalance mirrors broader challenges in organizational KM-particularly the difficulty of transforming individual outputs into collective and reusable knowledge. The findings suggest that GenAI is widely used to generate valuable artifacts (e.g., source code, documentation, test cases), but institutional mechanisms for dissemination and reuse have not evolved at the same pace.

Consequently, AI-generated knowledge often remains localized within specific projects or individual interactions, rather than being integrated into broader organizational memory. This highlights the need for new governance strategies, metadata standards, and validation workflows that ensure traceability, accountability, and collaborative learning across professional teams.

According to the SECI model, Externalization dominates current GenAI-related research, with 58 studies illustrating how tacit professional knowledge (e.g., natural language prompts or developer intuition) is converted into explicit artifacts such as source code and documentation. This reinforces the understanding of GenAI as a catalyst for articulating and formalizing expertise, bridging the gap between personal experience and shared, reusable artifacts. However, the scarcity of studies addressing Socialization and Internalization suggests that collective learning and experiential knowledge absorption are still underexplored dimensions.

The limited attention to these quadrants indicates that, while GenAI enhances content production, it has not yet been systematically leveraged to foster team learning, mentoring, or reflective knowledge exchange-aspects that are central to sustainable capability building and long-term organizational learning in software companies.

Emerging Research Directions

Overall, the mapping reveals a paradox: while GenAI amplifies knowledge production and reuse, it does not inherently guarantee knowledge integration or learning. The current scenario depicts software engineers producing more artifacts at a faster pace, yet not necessarily generating deeper organizational insight or sustained knowledge retention.

Thus, the integration of GenAI into SE appears to be pragmatic but not yet transformative. The technology primarily augments existing workflows rather than reshaping them, serving as a bridge between tacit and explicit knowledge. Future research should therefore focus on exploring how GenAI can evolve from a tool of individual productivity to a strategic enabler of collective and organizational knowledge in professional software engineering environments.

6.1 Research Opportunities

Based on the gaps and trends identified across the RQs, several opportunities for future investigation emerge:

- **Development of conceptual and methodological frameworks:** The limited number of studies proposing new frameworks (RQ1) suggests a need for theoretical grounding that can guide the integration of GenAI in Software Engineering and KM practices.
- **Broader adoption and analysis of open models:** Despite the availability of open-weight LLMs, their presence in the analyzed literature remains low (RQ2). Research could explore the potential of these models for academic reproducibility, cost reduction, and ethical transparency.
- **Alignment with organizational knowledge strategies:** There is little evidence of systematic alignment between GenAI use and formal KM strategies. Investigating how GenAI can support organizational learning, knowledge retention, and long-term value creation represents an open challenge.
- **Expanding focus beyond construction activities:** Most current studies target source code generation and related tasks (RQ3), leaving areas such as architecture, requirements management, and project planning underexplored.
- **Enhancing knowledge dissemination:** The relatively low attention given to knowledge sharing and dissemination (RQ4) indicates a need for strategies and tools that facilitate reuse, generalization, and organizational integration of AI-generated outputs.
- **Greater attention to underrepresented SECI modes:** Findings from RQ5 suggest that *Socialization* and *Internalization* are rarely addressed. These modes involve deep learning, mentoring, and intuition-building, which remain poorly understood in the context of GenAI.

6.2 Threats to the validity

In this section, we discuss potential limitations and biases in our mapping study, which may impact the interpretation and generalization of results. We discuss the validity concerning the four groups of common threats to validity [58].

Construct Validity: A potential threat is the decision not to apply backward and forward snowballing techniques. While these could improve comprehensiveness, we chose not to apply them for two main reasons: (i) most selected studies were published in 2024 and 2025, limiting the usefulness of forward snowballing due to the short citation window; and (ii) backward snowballing would likely retrieve older studies not aligned with the current state of GenAI research in Software Engineering. Additionally, we used Scopus, a comprehensive and high-quality multidisciplinary database, which returned 1264 studies. After applying strict inclusion and exclusion

criteria, 97 studies were selected, providing a recent and sufficiently broad overview of the domain.

Internal Validity: We followed a predefined, validated protocol, tested through a control group of relevant studies and refined iteratively. Screening, extraction, and classification were collaboratively distributed among authors, with regular meetings to resolve issues and exchange papers, ensuring multiple perspectives and reducing individual bias. Inspired by agile practices, this process ensured transparency and alignment throughout the review.

External Validity: Only English-language publications from Scopus were considered. While Scopus is one of the most comprehensive databases in Computer Science, this may have excluded relevant studies in other languages or gray literature. Despite this, the methodological rigor and the diversity of selected studies strengthen external validity. Future studies may broaden the scope by including additional sources.

Conclusion Validity: Some studies may have lacked methodological detail or omitted aspects relevant to our RQs. To address this, we used a conservative classification approach based only on clearly stated information. To support reproducibility, we made all data and analysis artifacts publicly available in a replication package.

7 Related Work

Before conducting the mapping study presented in this paper, we performed a tertiary study to investigate whether there were existing secondary studies addressing the same research topic. Tertiary studies are reviews that focus exclusively on secondary studies (Systematic Literature Reviews or Systematic Mappings) [34].

To conducted out the tertiary study, we used the same search string created for the mapping study presented in Table 2, and followed the approach for retrieving secondary studies proposed by [59]. The following search string was created:

("Chat GPT" OR "ChatGPT" OR "chatbot" OR "copilot" OR "generative AI" OR "generative artificial intelligence") AND ("software engineering" OR "computer science" OR "information technology" OR "software develop") AND ("systematic literature review" OR "systematic review" OR "systematic mapping" OR "mapping study" OR "systematic literature mapping" OR "literature review")*

We used Scopus as the search engine, and the string was applied in three metadata fields (title, abstract, and keywords), which returned 85 results. After analyzing these studies, we found that 53 of them were indeed secondary studies. However, none of the retrieved results addressed the specific objective of our study. In other words, none of them focused on systematic reviews or mapping studies about the use of GenAI in KM processes within organizations operating in the Software Engineering domain.

Since no secondary studies were found that simultaneously addressed the combined themes of Generative Artificial Intelligence (GenAI), Knowledge Management (KM), and Software Engineering, we chose to analyze secondary studies focusing exclusively on the relationship between GenAI and Software Engineering. This analysis revealed a diversity of research interests among the authors: 28 studies (56.0%) focused on educational applications, 7 studies (13.2%) centered on the analysis or development of chatbots, and 5 studies (9.4%) concentrated on software development activities, particularly programming techniques and practices.

These findings highlight a relevant and recent body of secondary studies concerning GenAI and Software Engineering, especially from 2019 onward. However, it is important to note that when the scope is narrowed to include studies that explicitly address Knowledge Management, no secondary studies have been identified to date.

8 Conclusion

This study aimed to analyze how GenAI has been applied in KM processes within the Software Engineering domain. A systematic mapping study was conducted, reviewing 97 primary studies. The analysis highlights five key findings: (i) research in this area remains focused on the practical use of existing tools, especially ChatGPT and GitHub Copilot, with few methodological or theoretical contributions; (ii) OpenAI-based solutions dominate both tools and models, indicating technological concentration; (iii) GenAI applications are mostly centered on Software Construction tasks, particularly code generation, with limited attention to project management, architecture, and quality assurance; (iv) regarding KM, GenAI primarily supports knowledge capture/creation and application, while sharing and dissemination remain underexplored; and (v) in terms of knowledge conversion, GenAI facilitates mainly externalization (tacit-to-explicit knowledge), with little focus on other SECI modes such as socialization and internalization.

The observed emphasis on tool usage and code-level automation suggests that practitioners are already leveraging GenAI to enhance productivity and reduce repetitive effort. However, without mechanisms for documentation, validation, and team-level knowledge sharing, there is a risk that AI-generated outputs become isolated artifacts rather than integrated organizational assets.

For researchers, these findings emphasize the need to expand inquiry beyond short-term productivity gains and explore the broader implications of GenAI for how software teams learn, collaborate, and evolve. There is also an opportunity to propose hybrid approaches that combine technical solutions with KM-oriented practices, ensuring that GenAI adoption supports both individual efficiency and collective capability building.

This research also reveals opportunities for future investigation, including exploring the role of GenAI in strategic and managerial software activities, developing mechanisms for disseminating AI-generated knowledge across organizations, and encouraging the use of open models to reduce dependence on proprietary solutions. Future work will refine the search protocol, extend the temporal and source scope, apply backward and forward snowballing, and conduct surveys with industry professionals to validate and expand the findings. In addition, methodological triangulation will be applied to strengthen the analysis and support the creation of practical guidelines for integrating GenAI into KM practices in Software Engineering.

SUPPLEMENTARY MATERIAL

Download all supplementary files included with this article:
https://doi.org/10.5281/zenodo.16372371

ACKNOWLEDGMENTS

Julia B. O. Fantini acknowledges financial support from the Institutional Scholarship Program in Scientific Initiation (PIBIC–UTFPR). Katia Romero Felizardo is supported by CNPq (Grant 302339/2022-1). All authors contributed to all activities related to the submission.

REFERENCES

[1] Ömer Aydın and Enis Karaarslan. Is chatgpt leading generative ai? what is beyond expectations? *Academic Platform Journal of Engineering and Smart Systems*, 11(3):118–134, 2023.

[2] Rishi Bommasani, Barret Zoph, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[3] Erik Brynjolfsson, Danielle Li, and Lindsey Raymond. Generative ai at work. *The Quarterly Journal of Economics*, page qjae044, 2025.

[4] Grant Cooper. Examining science education in chatgpt: An exploratory study of generative artificial intelligence. *Journal of science education and technology*, 32(3):444–452, 2023.

[5] Viriya Taecharungroj. "what can chatgpt do?" analyzing early reactions to the innovative ai chatbot on twitter. *Big Data and Cognitive Computing*, 7(1):35, 2023.

[6] John Aldrich. Generative ai and organizational knowledge: A review of current trends and future directions. *Journal of Knowledge Management*, 27(8):1756–1772, 2023.

[7] Ikujiro Nonaka and Hirotaka Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* Oxford University Press, 1995.

[8] Christoph Ziegler, Ute Schmid, and Martin Kramer. Productivity evaluation of github copilot as a pair-programmer in an introductory programming course. arXiv preprint arXiv:2210.14135, 2022.

[9] Prithvi Vaithilingam, Alaa Dakhel, and Michael Pradel. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*, pages 1–13. ACM, 2022.

[10] Daniel Russo. Navigating the complexity of generative ai adoption in software engineering. *ACM Transactions on Software Engineering and Methodology*, 33(5):1–50, 2024.

[11] David Sobania, Simon Krieghoff, Dennis Trautsch, Jens Grabowski, and Lennart Wermke. An empirical study on the usage of github copilot. In *Proceedings of the 45th International Conference on Software Engineering (ICSE)*. IEEE/ACM, 2023.

[12] Rémi Perrier, Xi Zhang, and Li Chen. Ai and software engineering: Ethical and practical considerations. *Software Engineering Notes*, 48(2):1–6, 2023.

[13] Xiaoyuan Lyu, Xiaohan Wang, Ziyang Li, Hongyu Yang, Hongyu Zhang, and David Lo. Gpt for se: Opportunities and pitfalls in adopting generative language models in software engineering. *arXiv preprint arXiv:2305.04639*, 2023.

[14] Yuxuan Zhang, Siqi Sun, Philip S. Yu, and et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[15] Tom B. Brown, Benjamin Mann, Nick Ryder, and et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[17] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Bhavani Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[18] D.E. O'Leary and R. Studer. Knowledge management: an interdisciplinary approach. *IEEE Intelligent Systems*, 16, No. 1, 2001.

[19] K. Dalkir. *Knowledge Management in Theory and Practice.* Elsevier, USA, 3 edition, 2017.

[20] K. M. Wiig. *Knowledge Management Foundations.* Schema Press, Limited, USA, 1 edition, 1994.

[21] M. H. Meyer and M. H. Zack. The design and development of information products. *MIT Sloan Management Review*, 37:43, 1996.

[22] M. McElroy. The knowledge life cycle. In *ICM Conference on KM*, Miami, USA, 1999. ICM.

[23] Wendi R Bukowitz and Ruth L Williams. *The knowledge management fieldbook.* Financial Times/Prentice Hall, Hoboken, EUA, 1 edition, 2000.

[24] Bianca Minetto Napoleão, Érica Ferreira de Souza, Glauco Antonio Ruiz, Katia Romero Felizardo, Giovani Volnei Meinerz, and Nandamudi Lankalapalli Vijaykumar. Synthesizing researches on knowledge management and agile software development using the meta-ethnography method. *Journal of Systems and Software*, 178:110973, 2021.

[25] Daniela Alves, Deivid Smek, Érica Souza, Katia Felizardo, and Nandamudi Vijaykumar. Updating a systematic literature review on knowledge management diagnostics in software development organizations. In *Anais do XXIII Simpósio Brasileiro de Qualidade de Software, Bahia/BA*, page 125–135, Porto Alegre, RS, Brasil, 2024. SBC.

[26] Sheikh Abdulaziz Fahad, Said A Salloum, and Khaled Shaalan. The role of chatgpt in knowledge sharing and collaboration within digital workplaces: a systematic review. *Artificial intelligence in education: The power and dangers of ChatGPT in the classroom*, pages 259–282, 2024.

[27] Tajinder Kumar, Vishal Garg, Sachin Lalar, and Rajinder Kumar. Measuring impact of generative ai in software development and innovation. In *International Conference on Entrepreneurship, Innovation, and Leadership*, pages 57–67. Springer, 2023.

[28] Juliana dos Santos, Guilherme Augusto Martins, Érica de Souza, Katia Felizardo, and Giovani Meinerz. Perceptions of knowledge management in brazilian software development companies. In *XXV Congresso Ibero-Americano em Engenharia de Software*, pages 233–247, 2022.

[29] P. Wendorff and D. Apshvalka. The knowledge management strategy of agile software development. pages 607–614, Univ. of Limerick, Ireland, 1998. 6th European Conference on Knowledge Management.

[30] G. A. Ruiz, P. R. da Silva, E. F. de Souza, K. R. Felizardo, G. V. Meinerz, and N. L. Vijaykumar. Knowledge management in agile testing teams: a survey. In *21st Ibero-American Conference on Software Engineering (CIbSE' 18)*, pages 1–8, 2018.

[31] Victor Cabrero-Daniel et al. Ai for agile development: a meta-analysis. *arXiv preprint arXiv:2305.08093*, 2023.

[32] Héctor Cornide-Reyes, Diego Monsalves, Esteban Durán, and Jorge Silva. Generative artificial intelligence in agile software development processes: A literature review focused on user experience. In *Lecture Notes in Computer Science*, volume 15787, pages 242–259. Springer, 2025.

[33] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *International Conference on Evaluation and Assessment in Software Engineering (EASE)*, pages 1–10, 2008.

[34] B.A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University and Durham University Joint Report, 2007. Technical Report EBSE 2007–001.

[35] D. Maplesden, E. Tempero, J. Hosking, and J.G. Grundy. Performance analysis for object-oriented software: A systematic mapping. *IEEE Transaction on Software Engineering*, 41(7):691–710, 2015.

[36] H. Zhang, B.A. Muhammad, and T. Paolo. Identifying relevant studies in software engineering. *Information and Software Technology*, 53(6):625–637, 2011.

[37] Lili Bo, Yuting He, Xiaobing Sun, Wangjie Ji, and Xiaohan Wu. A software bug fixing approach based on knowledge-enhanced large language models. In *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*, pages 169–179. IEEE, 2024.

[38] Nathan Hagel, Nicolas Hili, and Didier Schwab. Turning low-code development platforms into true no-code with llms. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, pages 876–885, 2024.

[39] Arianna Johnson. Here's what to know about openai's ChatGPT — what it's disrupting and how to use it. *Forbes*. Acessado em 18 de julho de 2025.

[40] Matthew B. Miles, A. Michael Huberman, and Johnny Saldaña. *Qualitative Data Analysis: A Methods Sourcebook.* SAGE Publications, Thousand Oaks, CA, 3rd edition, 2014.

[41] Suresh Babu Nettur, Shanthi Karpurapu, Unnati Nettur, and Likhit Sagar Gajja. Cypress copilot: Development of an ai assistant for boosting productivity and transforming web application testing. *IEEE Access*, 2024.

[42] IEEE Computer Society. *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) – Version 4.0.* IEEE, 2024. Accessed: July 2025.

[43] Mafura Uandykova, Laura Baitenova, Gulnar Mukhamejanova, Assel Yeleukulova, and Tolkyn Mirkassimova. Java coding using artificial intelligence. *Frontiers in Computer Science*, Volume 6 - 2024, 2024.

[44] Rebeka Tóth, Tamas Bisztray, and László Erdődi. Llms in web development: Evaluating llm-generated php code unveiling vulnerabilities and limitations. In *International Conference on Computer Safety, Reliability, and Security*, pages 425–437. Springer, 2024.

[45] Nenad Petrović, Samir Koničanin, and Suad Suljović. Chatgpt in iot systems: Arduino case studies. In *2023 IEEE 33rd International Conference on Microelectronics (MIEL)*, pages 1–4. IEEE, 2023.

[46] Saki Imai. Is github copilot a substitute for human pair-programming? an empirical study. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, pages 319–321, 2022.

[47] Christian Bird, Denae Ford, Thomas Zimmermann, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk, and Idan Gazit. Taking flight with copilot. *Communications of the ACM*, 66(6):56–62, 2023.

[48] Kimiz Dalkir. *Knowledge management in theory and practice.* routledge, 2013.

[49] Anh Nguyen-Duc and Dron Khanna. Value-based adoption of chatgpt in agile software development: A survey study of nordic software experts. In *Generative AI for Effective Software Development*, pages 257–273. Springer, 2024.

[50] Antonio Della Porta, Vincenzo De Martino, Gilberto Recupito, Carmine Iemmino, Gemma Catolino, Dario Di Nucci, Fabio Palomba, et al. Using large language models to support software engineering documentation in waterfall life cycles:

Are we there yet? In *CEUR WORKSHOP PROCEEDINGS*, volume 3762, pages 452–457. CEUR-WS, 2024.

[51] Michael J Muller, April Yi Wang, Steven I Ross, Justin D Weisz, Mayank Agarwal, Kartik Talamadupula, Stephanie Houde, Fernando Martinez, John T Richards, Jaimie Drozdal, et al. How data scientists improve generated code documentation in jupyter notebooks. In *IUI Workshops*, 2021.

[52] Lahbib Naimi, Abdeslam Jakimi, Rachid Saadane, Abdellah Chehri, et al. Automating software documentation: Employing llms for precise use case description. *Procedia Computer Science*, 246:1346–1354, 2024.

[53] Liuchun Zhan and Changjiang Huang. Research on computer intelligent chatgpt natural language processing system based on scientific knowledge graph. In *Proceedings of the 2024 International Conference on Machine Intelligence and Digital Applications*, pages 47–51, 2024.

[54] Daksh Chaudhary, Sri Lakshmi Vadlamani, Dimple Thomas, Shiva Nejati, and Mehrdad Sabetzadeh. Developing a llama-based chatbot for ci/cd question answering: A case study at ericsson. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 707–718. IEEE, 2024.

[55] Ningzhi Tang, Meng Chen, Zheng Ning, Aakash Bansal, Yu Huang, Collin McMillan, and Toby Jia-Jun Li. Developer behaviors in validating and repairing llm-generated code using ide and eye tracking. In *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 40–46. IEEE, 2024.

[56] Soham Patel, Kailas Patil, and Prawit Chumchu. Bhramari: Bug driven highly reusable automated model for automated test bed generation and integration. *Software Impacts*, 21:100687, 2024.

[57] Yuxin Qiu, Jie Hu, Qian Zhang, and Heng Yin. Calico: Automated knowledge calibration and diagnosis for elevating ai mastery in code tasks. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1785–1797, 2024.

[58] Xin Zhou, Yuqin Jin, He Zhang, Shanshan Li, and Xin Huang. A map of threats to validity of systematic literature reviews in software engineering. In *23rd Asia-Pacific Software Engineering Conferencees*, pages 153–160, 2016.

[59] Bianca M. Napoleão, Katia R. Felizardo, Érica F. de Souza, Fabio Petrillo, Sylvain Hallé, Nandamudi L. Vijaykumar, and Elisa Y. Nakagawa. Establishing a search string to detect secondary studies in software engineering. In *47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 9–16, 2021.