A Research Roadmap for Augmenting Software Engineering Processes and Software Products with Generative AI

A Research Roadmap for Augmenting Software Engineering Processes and Software Products with Generative AI

DOMENICO AMALFITANO, University of Naples "Federico II", Italy
ANDREAS METZGER*, Ruhr Institute for Software Technology (paluno), University of Duisburg-Essen, Germany
MARCO AUTILI, Università degli Studi dell'Aquila, Italy
TOMMASO FULCINI, Politecnico di Torino, Italy
TOBIAS HEY, Karlsruhe Institute of Technology (KIT), Germany
JAN KEIM, Karlsruhe Institute of Technology (KIT), Germany
PATRIZIO PELLICCIONE, Gran Sasso Science Institute (GSSI), Italy
VINCENZO SCOTTI, Karlsruhe Institute of Technology (KIT), Germany
ANNE KOZIOLEK, Karlsruhe Institute of Technology (KIT), Germany
RAFFAELA MIRANDOLA, Karlsruhe Institute of Technology (KIT), Germany
ANDREAS VOGELSANG, Ruhr Institute for Software Technology (Paluno), University of Duisburg-Essen, Germany

Abstract. Generative AI (GenAI) is rapidly transforming software engineering (SE) practices, influencing how SE processes are executed, as well as how software systems are developed, operated, and evolved. This paper applies design science research to build a roadmap for GenAI-augmented SE. The process consists of three cycles that incrementally integrate multiple sources of evidence, including collaborative discussions from the FSE 2025 "Software Engineering 2030" workshop, rapid literature reviews, and external feedback sessions involving peers. McLuhan's tetrads were used as a conceptual instrument to systematically capture the transforming effects of GenAI on SE processes and software products. The resulting roadmap identifies four fundamental forms of GenAI augmentation in SE and systematically characterizes their related research challenges and opportunities. These insights are then consolidated into a set of future research directions. By grounding the roadmap in a rigorous multi-cycle process and cross-validating it among independent author teams and peers, the study provides a transparent and reproducible foundation for analyzing how GenAI affects SE processes, methods and tools, and for framing future research within this rapidly evolving area.

*Corresponding author.

Authors' Contact Information: Domenico Amalfitano, domenico.amalfitano@unina.it, University of Naples "Federico II", Napoli, Italy; Andreas Metzger, andreas.metzger@paluno.uni-due.de, Ruhr Institute for Software Technology (paluno), University of Duisburg-Essen, Essen, Germany; Marco Autili, marco.autili@univaq.it, Università degli Studi dell'Aquila, L'Aquila, Italy; Tommaso Fulcini, tommaso.fulcini@polito.it, Politecnico di Torino, Turin, Italy; Tobias Hey, hey@kit.edu, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany; Jan Keim, jan.keim@kit.edu, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany; Patrizio Pelliccione, patrizio.pelliccione@gssi.it, Gran Sasso Science Institute (GSSI), L'Aquila, Italy; Vincenzo Scotti, vincenzo.scotti@kit.edu, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany; Anne Koziolek, anne.koziolek@kit.edu, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany; Raffaela Mirandola, raffaela.mirandola@kit.edu, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany; Andreas Vogelsang, andreas.vogelsang@uni-due.de, Ruhr Institute for Software Technology (Paluno), University of Duisburg-Essen, Essen, Germany.

ACM 1557-7392/2026/1-ART
https://doi.org/10.1145/3788879

CCS Concepts: • **Software and its engineering → Software development techniques**; • **Computing methodologies → Artificial intelligence**.

Additional Key Words and Phrases: Generative AI, Agentic AI, Foundation Models, Software Engineering, Software Engineering Process, Software Development Life Cycle, Software Product, Research Roadmap

1 Introduction

Software engineering (SE) is experiencing a transformation of unprecedented speed and scale, driven by the growing augmentation in software engineering *processes* and software *products* with Generative Artificial Intelligence (GenAI) [1, 2, 73]. Such GenAI-augmentation offers unprecedented opportunities for efficiently and effectively developing and operating novel kinds of software systems and applications, but at the same time challenges established principles, practices, tasks, activities and processes of software engineering [2, 7, 30, 88]. It requires rethinking existing software development lifecycle models (SDLC models), particularly questioning long-held assumptions about the roles of software engineers, the dynamics of hybrid human-AI teams, and the very nature of software artifacts, which in addition to code and data, now include GenAI models and prompts.

We systematically identify those challenges and opportunities by starting from a structuring of the main forms GenAI -augmentation in SE can take. This structuring consists of two dimensions that answer the following questions.

"What is augmented by GenAI?" This dimension creates a distinction between using GenAI to augment SE *processes* versus using GenAI to augment software *products*:

- GenAI-augmented SE *processes* mean that GenAI is used to automate software engineering activities and tasks. Examples range from requirements analysis, through automatic code completion and code generation, via test case generation and test case prioritization, to optimized CI/CD pipelines and continuous code refactoring and maintenance [2, 34, 39, 39, 79, 84, 89].
- GenAI-augmented software *products* mean that parts of the functionality of a software system or application are not explicitly programmed but are realized with GenAI. Such GenAI-augmented software products can generate novel content, power sophisticated conversational interfaces, or create adaptive user experiences [35, 60, 68, 95].

"How autonomous is the GenAI augmentation?" This dimension creates a distinction of whether GenAI plays a *passive* or an *active* role:

- In a *passive* role, GenAI does not have goals or act on its own. Instead, GenAI responds reactively to user prompts or inputs provided via an API. Or, in other words, the human-AI interaction is triggered by the human [35, 95]. A simple example is an FAQ chat interface included in a website.
- In an *active* role, GenAI possesses its own thread of control and makes (semi)autonomous decisions about which actions to perform and when [55, 56, 91]. This means GenAI not only processes information or generates content, but can also proactively make decisions and perform activities and tasks. Or, in other words, the human-AI interaction is triggered by AI. A typical manifestation of such an active role is Agentic AI, which refers to a collection of (semi)autonomous AI agents that jointly aim to achieve higher-level goals [41, 83, 98].

The intersection of these two dimensions results in four distinct forms of GenAI augmentation in SE. On the one hand, these four forms provide us with a clear and concise structure to analyze the state-of-the-art and provide a research roadmap for form-specific research challenges and opportunities. Hence, these four forms provide more rigor than loose terms such as "AI for SE", which may refer to using AI in any form to enhance SE processes, or "SE for AI", which may refer to extending established SE principles and techniques to the unique challenges posed by developing AI models or AI-augmented systems and applications. On the other hand, these

four forms help us achieve a more systematic and comprehensive coverage of GenAI augmentation in SE, thereby also allowing us to identify cross-form research challenges and opportunities.

Overall, we make the following main contributions to SE research:

- **Classification of GenAI Augmentation**: We provide a classification of GenAI in SE based on the aforementioned two dimensions, resulting in four distinct forms: GenAI Copilot, GenAIware, GenAI Teammate, and GenAI Robot.
- **Analysis of GenAI Effects:** We systematically capture the effects of GenAI on SE for each of the above forms, determining what GenAI enhances, reverses, retrieves from the past, and makes obsolete.
- **Comprehensive Research Roadmap:** We distill those findings into a detailed roadmap identifying form-specific and cross-cutting research challenges and opportunities. Key areas include prompt engineering, accountability, hybrid human-AI team dynamics, and the coordination of process-level versus product-level agents.
- **Predictions for 2030:** We conclude with ten strategic predictions for how SE may look like in the year 2030. This includes the death of manual coding for routine tasks and the shift from the role of SE developer to AI agent orchestrator.

We structure the remainder of the article as follows. In Section 2, we detail the methodology followed during our review and analysis. In Section 3, we elaborate on the four forms of GenAI augmentation and discuss related research roadmaps. In Sections 4 through 7, we present the state of the art and a detailed analysis of the impact of GenAI augmentation for each of the four forms. In Section 8, we distill a roadmap composed of a set of research challenges and opportunities. In Section 9 we discuss validity risks. In Section 10, we conclude the paper.

2 Methodology

This section presents the methodology followed to perform this study. We adopted the Design Science Research (DSR) approach [9, 38, 97], as our primary goal was to design a well-founded *artifact* that supports understanding and future development of the GenAI-driven Software Development Life Cycle (SDLC). In our case, the artifact is the *Roadmap for the SDLC in the GenAI era*, which was progressively derived through a sequence of McLuhan Tetrads capturing challenges, limitations, and opportunities observed in current software engineering practices. Figure 1 visualizes the overall DSR process, which was structured into three cycles. Each cycle is characterized by distinct tasks and produces specific outputs that progressively shape the final roadmap.

2.1 Cycle 1 – Initial Investigation

The first cycle was intended to create initial awareness and structure the problem space.

2.1.1 Awareness - 2030 Software Engineering Workshop @FSE2025. The inspiration and basis for this special issue article was the "2030 Software Engineering" workshop[1], co-located with the ACM SIGSOFT FSE 2025 conference (June 26–27, Trondheim, Norway), in which most of the authors of this article participated. The workshop adopted the format of "liberating structures"[2], leading to two full days of intensive discussions on topics including AI for software engineering, software engineering for AI, sustainable software engineering and quantum software engineering.

2.1.2 Solution - Structuring GenAI augmentation in SE. To facilitate a systematic discussion of the impact of GenAI augmentation, an initial structure was defined for the different forms of GenAI augmentation in SE. The structure was informed by discussions and insights from the FSE 2025 workshop and complemented by earlier

[1] https://conf.researchr.org/home/2030-se-2025
[2] https://www.liberatingstructures.com/

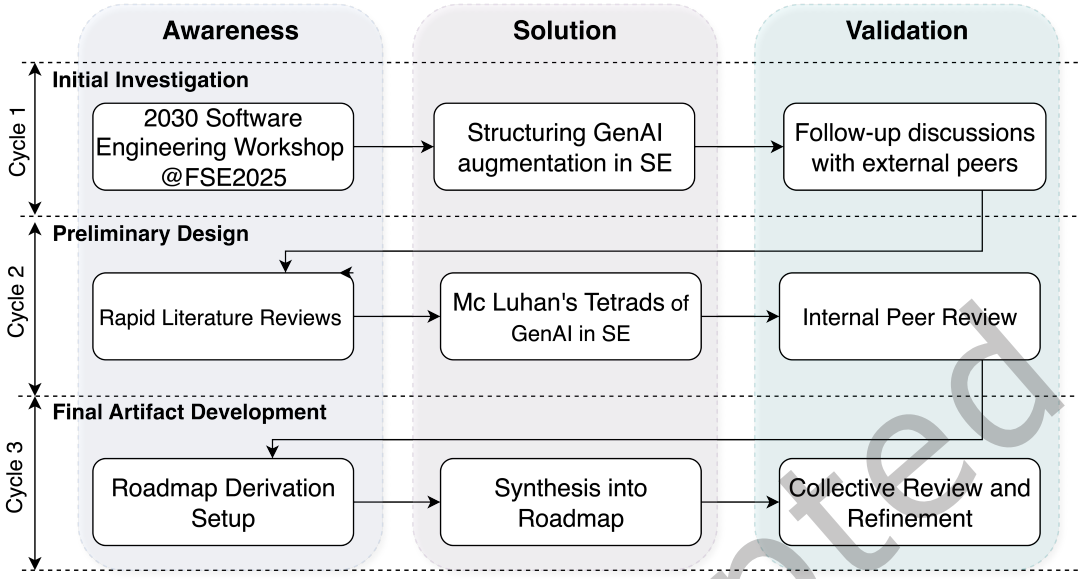

Fig. 1. The design science approach for building a roadmap on GenAI-augmented processes and products in SE

efforts to classify GenAI augmentation in SE, notably the taxonomy by CMU's Software Engineering Institute (SEI) in 2023 [71].

This initial effort resulted in a structure for GenAI augmentation in SE along the two dimensions into four distinct forms as introduced in Section 1. These four forms helped us analyze related roadmaps and understand how our article may refine these. Additionally, these four forms provided the structure for Cycle 2, where rapid literature reviews were conducted to collect evidence and characterize each of these four forms in detail.

2.1.3 Validation - Follow-up Discussions with External Peers. The resulting structure was validated through discussions with seven fellow SE professors from the Ruhr Institute for Software Technology (paluno[3]) during a two-day research retreat.

2.2 Cycle 2 – Preliminary Design

Cycle 2 focused on the collection of evidence through rapid literature reviews and the determination of the impact of GenAI augmentation using McLuhan tetrads.

2.2.1 Awareness - Rapid Literature Reviews. When the research focus is well defined and unambiguous, as in our case, rapid literature reviews (RLRs) constitute pragmatic evidence synthesis methods designed to deliver timely and resource-efficient insights compared to full-fledged Systematic Literature Reviews (SLRs) or Systematic Mapping Studies (SMSs) [14]. Unlike SLRs and SMSs, RLRs typically relax some methodological requirements, such as triangulation among multiple researchers and systematic validation of study selection and data extraction steps. Instead, these activities are often conducted by one or at most two researchers, which reduces the possibility of cross-checking but allows the review to be completed within a short time frame (usually one or two months). Despite this lower degree of formality, prior work in software engineering has shown that RLRs still provide valuable evidence to support decision-making and efficiently characterize emerging research domains [14, 65].

[3]https://paluno.uni-due.de/en/

We adopted this approach because, after the FSE workshop held at the end of June (see Section 2.1.1), only a few months remained to obtain an initial evidence-based overview of the four fundamental forms of GenAI augmentation in SE. Consequently, small teams of one or two authors conducted focused RLRs, each team addressing one fundamental form, and completed the activity within two months. This pragmatic choice allowed us to align the literature analysis with the design science process, ensuring that each fundamental form was grounded on an initial evidence base despite the short time frame. To provide additional transparency and replicability of the RLR process, we provide open access to the data of our literature reviews [17].

Research Protocol. All rapid literature reviews followed a shared baseline protocol, which was then specialized by each group to fit the specific characteristics of the form under investigation. The protocol consisted of the following steps:

(1) **Selection of bibliographic databases:** Relevant digital libraries were selected as sources for each form. In a pragmatic manner, and depending on the coverage of the topic, in addition to standard databases such as Scopus, Web of Science, IEEE Xplore, and the ACM Digital Library, we also used Google Scholar to capture recently published papers not yet indexed elsewhere.
(2) **Search string definition.** Search strings were constructed from form-specific constructs and iteratively refined by the responsible authors.
(3) **Inclusion and exclusion criteria (IC/EC):** Explicit criteria were defined to determine whether publications should be included or excluded, ensuring that only relevant contributions were retained. The following core inclusion and exclusion criteria (applied consistently across all reviews) were defined, while some groups complemented them with additional form-specific criteria:
    - **IC:** (i) studies employ techniques of generative GenAI (e.g., Large Language Models such as GPT, Generative Adversarial Networks, or Variational Autoencoders); (ii) studies explicitly address the SDLC and/or software development and operations processes.
    - **EC:** (i) studies not written in English; (ii) duplicated studies; (iii) studies authored by the same group without introducing new contributions.
(4) **Search execution:** The search strings were executed in the selected databases and the retrieved records aggregated.
(5) **Screening and selection:** Depending on the group, records were screened either in two phases (titles/abstracts followed by full-text) or directly at full-text level, by applying the IC/EC to identify the final set of primary studies.
(6) **Analysis and synthesis:** The selected publications were analyzed, and their relevant content synthesized into the entries of the tetrads, i.e., the constituents of each quadrant for the corresponding fundamental form.

2.2.2 Solution - McLuhan's Tetrads for GenAI augmentation in SE. Marshall McLuhan's Tetrad of media effects[4] provides a framework for analyzing and visualizing the effects of a technology by categorizing them into four interrelated dimensions as explained below. We chose McLuhan's Tetrads for our analysis, as they were successfully used in previous studies. In addition to being used during the FSE 2025 workshop (see Section 2.1.1) and its predecessor at FSE 2024 [74] to examine the impact of GenAI on SE, they were also used to examine the effects of LLMs on SE research [90].

For each form of GenAI, a preliminary McLuhan's Tetrad was created based on the analysis of the selected publications and insights from the FSE 2025 workshop (see Section 2.1.1), thereby synthesizing the findings. Each tetrad was independently developed by a team of authors, with no author contributing to more than one tetrad, to avoid bias and facilitate diversity of perspectives.

[4]https://en.wikipedia.org/wiki/Tetrad_of_media_effects

The four interrelated dimensions of a McLuhan's Tetrad are depicted in Figure 2. Together, these dimensions offer a holistic perspective on the consequences of technology adoption.

(1) "What does the technology **enhance** or intensify?" This aims to understand how the technology augments or amplifies certain capabilities. For example, the advent of the printing press intensified the dissemination of information, revolutionizing communication and education.
(2) "What does the technology **reverse** or flip into when pushed to its extreme?" This examines how certain technology, when pushed to its extreme, undergoes a reversal or transformation. As an example, the ubiquity of smartphones, offering constant connectivity, has the potential to reverse into isolation and disconnection.
(3) "What does the technology **retrieve** or recover from the past?" This aims to understand how far technology makes it possible to retrieve or recover concepts and ideas from the past. As an example, the advent of the Internet retrieved the decentralized and participatory nature of oral communication in the digital realm.
(4) "What does the technology make **obsolete** or displace?" This explores the obsolescence or displacement caused by the technology in question. As an example, the rise of television displaced radio as the primary source of news and entertainment.

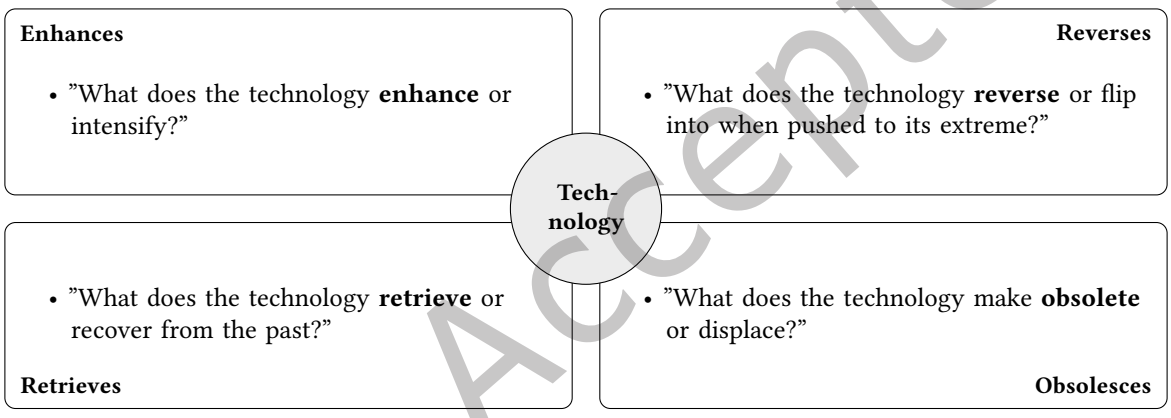

Fig. 2. A blank tetrad diagram.

2.2.3 Validation - Internal Peer Review. The preliminary results, including the structuring into the four forms, the findings of the rapid literature review, and the derived tetrads, were subjected to internal peer reviews between the authors. Each tetrad was validated by the authors who had not participated in the derivation of the respective tetrad, ensuring a collective assessment and reducing the risk of confirmation bias. Feedback was collected to refine the structure, clarify ambiguities and improve the overall consistency of intermediate results.

2.3 Cycle 3 – Final Artifact Development

Cycle 3 focused on the development of the final artifact, that is, the roadmap for the augmentation of GenAI in SE, and its collective validation.

2.3.1 Awareness - Roadmap Derivation Setup. In this phase, two authors, who were not involved in the construction of individual tetrads, analyzed the consolidated tetrads from Cycle 2 with the aim of synthesizing an overall research roadmap on GenAI augmentation in SE. Their external position with respect to the earlier cycles was intended to ensure a fresh and unbiased view on the consolidated tetrads.

2.3.2 Solution - Synthesis into Roadmap. To synthesize the roadmap, the two authors systematically analyzed all validated tetrads, identifying transversal patterns, recurring themes, and complementary insights. The results of this synthesis were distilled into a comprehensive roadmap that covers the form-specific and cross-form research challenges and opportunities.

2.3.3 Validation - Collective Review and Refinement. The roadmap draft was subjected to a validation process involving all remaining authors. Feedback was provided asynchronously through comments and revisions and was further discussed in three one-hour meetings. This iterative process allowed for clarifications, refinements, and resolution of disagreements, ultimately leading to the final version of the roadmap.

3 Preliminaries

Resulting as an outcome of Cycle 1 of our methodology (see Section 2.1), we provide the foundations for a systematic discussion of the impact of GenAI augmentation in SE by elaborating on the four different forms of GenAI augmentation, and discussing related research roadmaps.

3.1 Structuring of GenAI in SE

As introduced in Section 1, we define the following two dimensions:

- **"What is augmented by GenAI?"**, distinguishing between GenAI-augmented SE *processes* (meaning that GenAI is used to automate software engineering activities and tasks) and GenAI-augmented software *products* (meaning that parts of the functionality of a software system or application are not explicitly programmed, but are implemented with GenAI).
- **"How autonomous is the GenAI augmentation?"**, distinguishing between *passive* roles (meaning that the human-AI interaction is triggered by the human [35, 95] and that the behavior of GenAI is determined by the methods invoked upon the GenAI model) and *active* roles (meaning that the human-AI interaction is triggered by GenAI and that GenAI has its own thread of control and makes (semi)autonomous decisions about which actions to perform and when; e.g., Agentic AI).

The intersection of these two dimensions forms a 2x2 matrix, as shown in Table 1, which defines four distinct forms of GenAI augmentation in SE.

Table 1. Structuring of GenAI in SE: Four distinct forms of GenAI augmentation

		"What is augmented by GenAI?"	
		Process	Product
"How autonomous is the GenAI augmentation?"	Passive Role	**GenAI Copilot**	**GenAIware**
	Active Role	**GenAI Teammate**	**GenAI Robot**

To be able to refer to these forms succinctly in this article, we use the following labels throughout the paper:

GenAI Copilot: GenAI is used as a tool – either standalone or integrated into an IDE or CI/CD pipeline – to automate various SE tasks. Here, one typically can find solutions from the field of Automated Software Engineering. For the label GenAI Copilot, we took inspiration from Github Copilot [22] and many other AI/GenAI Copilots available. Examples of tasks supported by GenAI Copilot include requirement elicitation [52], code generation [43], testing [92], and program repair [104].

GenAIware: GenAI is used to realize software functionality that otherwise would be impossible to realize or require a significant effort to realize [95]. This means that GenAI is not used to generate actual code, but that

the GenAI model is invoked from the code to perform specific computations. For the label GenAIware, we took inspiration from FMware, which refers to "the type of software that uses foundation models (FMs), such as Large Language Models (LLMs), as one of its building blocks" [35]. In this sense, the label "GenAIware" generalizes from FMware to encompass a wider range of GenAI models. An example of GenAIware is a software system that leverages large language models (LLMs) to summarize instructional video transcriptions, thus offering support to teachers and students [16]. Here, the LLM prompt is far more than a simple text input – it is a meticulously engineered part of the codebase. Another example is augmenting an online store with a chatbot to guide customers in finding products that fit their needs.

GenAI Teammate: GenAI acts as an agent that proactively participates in the software development process in one or more roles. GenAI Teammates are (semi)autonomous, goal-driven agents that collaborate with human software engineers in real-world workflows. For the label GenAI Teammate, we took inspiration from a recent post from the World Economic Forum[5]. Examples are autonomous coding agents, which actively initiate, review and evolve code on a scale as part of open source ecosystems [55].

GenAI Robot: GenAI acts as a (semi)autonomous, goal-driven agent to deliver parts of the functionality of the software system. For the label GenAI Robot, we took inspiration from the general definition of a robot as a "machine [...] capable of carrying out a complex series of actions automatically"[6]. As a simple example, a GenAI Robot can browse the Web and make online purchases on behalf of a user: compare prices, select items, and complete checkouts [29]. As a more complex example, a process-aware information system that facilitates the on-boarding of new suppliers as part of a procurement process may be implemented via GenAI Robots [27]. In such a system, there may be a *Buyer* GenAI Robot and different *Supplier* GenAI Robots. The Buyer GenAI Robot identifies the potential Supplier GenAI Robots and issues a request for quotes, which is then answered by the Supplier GenAI Robots. Based on these answers, the Buyer GenAI Robot can then select a suitable supplier.

Note that while we used the 2x2 matrix to define a set of four distinct forms, in a concrete software systems more than one form may be present simultaneously. For instance, a highly advanced GenAI Copilot could also exhibit some proactive behavior of a GenAI Teammate.

3.2 Related Research Roadmaps

Here we discuss how our article relates to existing efforts concerning research roadmaps on GenAI in SE. On the one hand, we discuss recent roadmap papers (journal articles and conference papers published since 2024). On the other hand, we assess the scope of research workshops that took place since 2024 at major SE conferences, as they often feature a set of very timely forward-looking papers based on the participants' contributions and onsite discussions.

Concerning related roadmap articles, the first TOSEM special issue on "2030 Roadmap for Software Engineering" particularly stands. This special issue presents the results of intensive 2-day discussions at the 2030 Software Engineering Workshop, co-located with FSE 2024 [74]. The presented roadmap was intended to serve as a living body to be refined based on follow-up workshops and updated during a series of forthcoming TOSEM special issues. Our article presents such an update for one of the roadmap's seven themes: "artificial intelligence for software engineering" [2]. This theme was covered in the 2025 TOSEM special issue by nine contributed articles. Six of those discuss the impact of AI on the SE process (thereby providing initial directions on GenAI Copilots and GenAI Teammates). The remaining three articles deal with the integration of AI into systems and applications (thereby providing initial directions on GenAIware and GenAI Robots). In addition to the aforementioned special issue articles, Table 2 lists relevant roadmap articles on GenAI in SE over the past two years and identifies which of the four forms of GenAI augmentation they addressed. The list of roadmap articles indicates a very strong focus

[5]https://www.weforum.org/stories/2025/01/why-you-should-think-of-ai-as-a-teammate-not-a-tool-when-building-a-better-future/
[6]https://en.wikipedia.org/wiki/Robot

Table 2. Coverage of GenAI augmentation in SE roadmap papers: ++ = main focus of article, + = secondary focus

Year	Journal	Title	Copilot	AIware	Teammate	Robot
2025	Autom. Softw. Eng.	Future of software development with generative AI [84]	++			
	Applied Sc.	AI-driven innovations in software engineering: a review of current practices and future directions [3]	++			
	Inf. Softw. Technol.	Copiloting the Future: How Generative AI Transforms Software Engineering [7]	++		+	
	SP&E	Generative Artificial Intelligence for Software Engineering—A Research Agenda [70]	++			
	SSRN	Generative AI for Software Architecture, Applications, Trends, Challenges, and Future Directions [23]	++		+	
	TOSEM-SI	2030 Roadmap for Software Engineering [74]	++	++	+	+
	arXiv	Challenges and Paths Towards AI for Software Engineering [33]	++		+	
	arXiv	From LLMs to LLM-based Agents for Software Engineering: A Survey of Current, Challenges and Future [44]	++		++	
2024	TAAS	Generative AI for Self-Adaptive Systems: State of the Art and Research Roadmap [57]		++		

on GenAI Copilots, or in other words automated SE, while only few address the other forms or even multiple forms at the same time. Some of the recent roadmap articles touch on the aspect of autonomous GenAI, but mainly focus on GenAI Teammates.

Concerning related roadmap workshops, Table 3 lists the relevant ones[7] on GenAI in SE of the past two years held at the three top-tier (A*) SE conferences and identifies which of the four forms of GenAI augmentation they addressed. The list of workshops allows us to make the following high-level observations. First, there is a clear increase between 2024 and 2025 in the number of workshops dedicated to the topic of GenAI, indicating the increasing relevance of GenAI in SE. Second, similar to the roadmap articles, most workshops focus on GenAI Copilots. Third, some of the recent workshops have started to cover the aspect of autonomous GenAI, but mainly focus on GenAI Teammates.

As a major difference from the above roadmap articles and forward-looking workshops, our roadmap covers all four forms, thus providing both individual and cross-form research challenges and opportunities.

4 Impact of GenAI Copilots in SE

This section presents the outcomes of Cycle 1 conducted on the GenAI Copilot form. As introduced in Section 3.1, a GenAI Copilot is a passive element (such as a component, service, or tool) that supports human software engineers in performing development tasks. We describe how this review was designed and executed using a refinement of the general protocol defined in Section 2.2 and discuss the McLuhan tetrad derived from the collected evidence.

[7]Note that for AgenticSE@ASE'25 and Ex-ASE@ASE'25 there was no program available at time of writing.

Table 3. Coverage of GenAI augmentation in SE workshops: ++ = main focus of workshop, + = secondary focus

Conference	Workshop		Copilot	AIware	Teammate	Robot
ICSE'25	AIOps	AI for Cloud Service		++		
	APR	Automated Program Repair		++		
	BotSE	Bots in SE	++	+		
	DeepTest	Deep Learning <> Testing		++	++	
	LLM4Code	Large Language Models for CODE	++			
	NLBSE	Natural Language Based SE	++			
	NSE	Neuro-Symbolic SE	+			
	RAIE	Responsible AI Engineering	++			
	RAISE	Requirements Eng. for AI-Powered SW	++			
	IWSIB	Software-Intensive Business		+		
ICSE'24	APR	Automated Program Repair	+			
	DeepTest	Deep Learning <> Testing		+		
	FTW	Flaky Tests Workshop		+		
	Intense	Interpretability, Robustness, and Benchmarking in Neural SE	++			
	IWSIB	Software-Intensive Business		+		
	LLM4Code	Large Language Models for Code	++			
	NLBSE	Natural Language Based SE	++			
	RAIE	Responsible AI Engineering	+	+		
	SATrends	New Trends in Software Architecture	+			
FSE'25	2030 SE	2030 Software Engineering	++	++	++	+
	AI IDE	Artificial Intelligence for Integrated Development Environments	++			
	AI-SDLC	Envisioning the AI-Augmented Software Development Life Cycle	++		++	
	BI4LLMC	Benchmark Infrastructure for LLMs for Code	++			
	Human AISE	Human-Centered AI for SE	++	++		
	LLinMAP	Large Language Model-Oriented Empirical Research	++	+		
	LLMApp	LLM App Store Analysis		+	+	
	ResponsibleSE	Engineering Responsible SE	+			
FSE'24	2030 SE	2030 Software Engineering	++	++	+	+
	AISM	AI for Software Modernization	++			
ASE'24	Intelligent SE	Intelligent Software Engineering	++			
	MAS-GAIN	Multi-Agent Systems using Generative Artificial Intelligence for ASE			++	
	ASYDE	Automated and verifiable Software syStem Development		++		

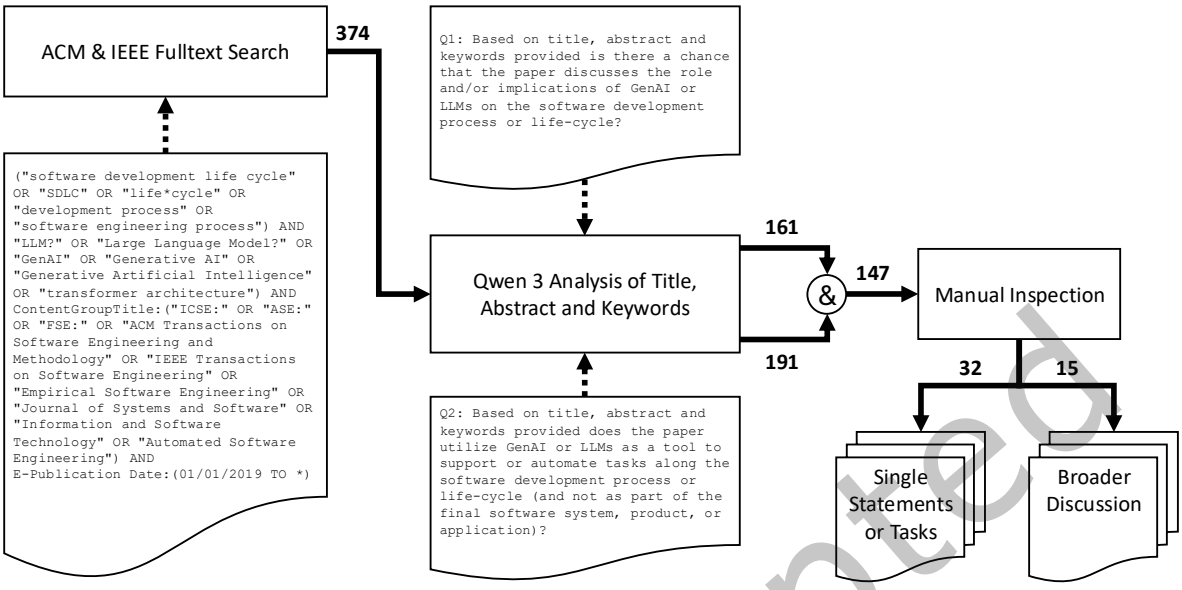

Fig. 3. Identifying relevant publications concerning GenAI Copilots

4.1 Rapid Literature Review

To provide an overview of the implications and impacts of GenAI Copilots on SE as identified in the literature, we performed a rapid literature review as explained in Section 2.2.

4.1.1 Review Approach. The procedure we followed to perform the rapid literature review for GenAI Copilots is depicted in Figure 3.

As this category of GenAI has attracted the most research focus in software engineering to date, we limited our search to top-tier software engineering conferences and journals (ICSE, ASE, FSE, TOSEM, TSE, EMSE, JSS, IST, ASEJ). We restricted the search to the years 2019–present, as the first Transformer-based models (BERT and GPT) were released in 2018, requiring time for adoption by the SE community. To ensure comprehensive coverage, we opted for full-text searches using ACM and IEEE as search engines, as we initially found that many papers only discuss the impact on SE in the body of the paper and not in the title, abstract, or keywords. We narrowed the scope to papers explicitly mentioning the SDLC, development process, or software engineering process in conjunction with LLMs, GenAI, or transformer architecture. These criteria are reflected in the search string[8] presented in Figure 3.

To further streamline the analysis process, we leveraged Qwen3[9] to analyze the titles, abstracts, and keywords of the retrieved publications. We asked Qwen3 to answer two questions about the publication that reflect the scope of our search: (Q1) Whether the paper explicitly discusses the role and/or implications of GenAI on the SDLC; (Q2) Whether the paper utilizes GenAI or LLMs as a tool to support or automate tasks along the SDLC. Q1 focuses on the publication's scope, while Q2 excludes GenAIware or GenAI Robot publications, as these are addressed in the chapters below. For each question, Qwen3 provided a yes/no answer along with its reasoning[10].

[8]The exact search strings can be found in Section A.1
[9]https://ollama.com/library/qwen3:8b
[10]The exact prompts can be found in Section B.1

Table 4. Relevant publications concerning GenAI Copilots

Paper	Authors	Year
Future of software development with generative AI [84]	Sauvola et al.	2024
A disruptive research playbook for studying disruptive innovations [87]	Storey et al.	2024
Large language models for software engineering: a systematic literature review [39]	Hou et al.	2024
Software testing with large language models: Survey, landscape, and vision [92]	Wang et al.	2024
Formal requirements engineering and large language models: A two-way roadmap [26]	Ferrari and Spoletini	2025
Copiloting the future: How generative AI transforms software engineering [7]	Banh et al.	2025
Challenges and opportunities for generative AI in software engineering: a managerial view [79]	Rico and Öberg	2025
What do professional software developers need to know to succeed in an age of Artificial Intelligence? [47]	Kam et al.	2025
AI in the software development lifecycle: Insights and open research questions [34]	Guimaraes and Nascimento	2025
The future of AI-driven software engineering [89]	Terragni et al.	2025
From today's code to tomorrow's symphony: The AI transformation of developer's routine by 2030 [76]	Qiu et al.	2025
From triumph to uncertainty: The journey of software engineering in the AI era [64]	Mastropaolo et al.	2025
Automatic programming: Large language models and beyond [62]	Lyu et al.	2025
The current challenges of software engineering in the era of large language models [30]	Gao et al.	2025
Artificial intelligence for software engineering: The journey so far and the road ahead [2]	Ahmed et al.	2025
Towards Automated Knowledge Management in the Software Life Cycle [48]	Keim et al.	2025

The publications positively evaluated by Qwen3 for both questions were then manually inspected by two authors (split equally).

4.1.2 Review Results. The initial search in ACM and IEEE resulted in 374 publications. Qwen3 classified 161 of these as discussing the role or implications of GenAI on the SDLC and 191 as utilizing GenAI as a tool to support or automate tasks along the SDLC. Of these, 147 met both criteria and were manually inspected. During the manual inspection process, the authors identified that most publications only mention SDLC as motivation or in the related work, but do not describe the implications of GenAI on it (100 articles). Furthermore, many publications made only single statements about the impact or role of GenAI in SDLC, or single tasks of the SDLC that are improved (32 articles). Only a few publications provided a broader discussion of the implications and role of GenAI Copilots on the SDLC (15 papers, see Table 4).

The latter two categories were used to inform our McLuhan's tetrad (see Section 4.2), either by deriving insights from the single statements or analyzing the full discussion in the 15 other papers. Furthermore, we included our proposal for the FSE 2025 workshop on "Automated Knowledge Management in the Software Life Cycle" [48], as it was one of the starting points for this contribution and fits into the category GenAI Copilot.

4.2 McLuhan's Tetrad

Informed by our rapid literature review, we interpret each quadrant of the tetrad in Figure 4 and map the relevant findings of previous work into these four dimensions.

4.2.1 Enhances.

- *Requirements*: Ferrari and Spoletini [26] describe how the *Requirements* phase can be enhanced by LLMs through improved extraction, analysis, verification, and validation of requirements. Their two-way roadmap demonstrates the mutual influence between formal requirements engineering and advances in GenAI. Additionally, Rico and Öberg [79] highlight enhancements in the generation and refinement of requirements.

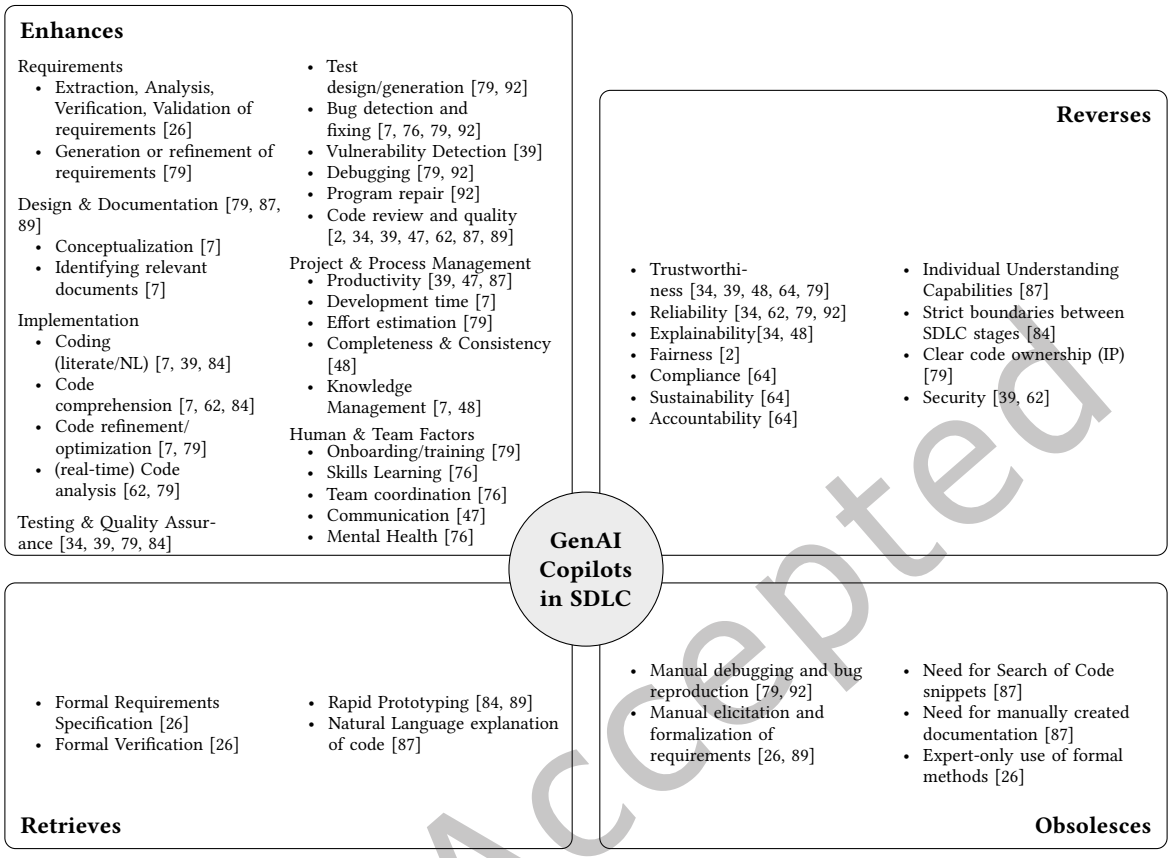

Fig. 4. McLuhan's tetrad derived from the rapid literature review of GenAI Copilot publications.

- *Design & Documentation*: GenAI Copilots have been reported to improve activities in *Design & Documentation* by enabling the automated creation of context-sensitive and stakeholder-specific artifacts, thereby reducing manual workload and ensuring that design intent is preserved throughout project iterations (Rico and Öberg [79], Terragni et al. [89]). Also, Storey et al. [87] mention in their tetrad for code generation, that documentation is enhanced by the introduction of GenAI Copilots. Furthermore, Banh et al. [7] describe how GenAI Copilots support conceptualization and provide contextual knowledge, such as relevant documents.
- *Implementation*: GenAI Copilots also enhance activities in the *Implementation* phase. For instance, coding itself is enhanced [7, 39, 84] through the abilities of the GenAI Copilots to translate natural language descriptions into functional code, enabling developers to work more efficiently and reducing the cognitive gap between design intent and implementation. For improving code comprehension [7, 62, 84], these tools can provide explanations of unfamiliar code. Additionally, in the refinement and optimization of code [7, 79], the literature identifies potential enhancements, such as GenAI Copilots suggesting performance improvements, cleaner syntax, and better algorithmic solutions. Code analysis [62, 79] is elevated with AI-driven insights that catch errors, suggest alternatives, and flag security vulnerabilities as the code is written (Real-time).

- *Testing & Quality Assurance*: GenAI Copilots also play a transformative role in *Software Testing & Quality Assurance* by improving the efficiency, coverage, and intelligence of testing and debugging processes [34, 39, 79, 84]. Test design and generation [79, 92] is enhanced by the ability to derive test cases from more diverse sources, such as documentation or even natural language requirements. GenAI Copilots also improve bug detection and fixing [7, 62, 84], by analyzing code context to locate faults and propose fixes, reducing the time developers spend on manual debugging. Vulnerability Detection [39] is enhanced by automatically identifying potential security risks based on learned patterns and best practices, even in complex codebases and potentially real-time. GenAI Copilots support or even enable automated debugging [79, 92] by explaining runtime errors, tracing root causes, and suggesting corrective actions in real-time. When bugs or vulnerabilities are found, GenAI Copilots can automatically generate fix suggestions or patches, enabling automated program repair [92]. In general, the introduction of GenAI Copilots can enhance code quality [2, 34, 47, 87, 89], especially by automating code reviews [39, 62], flagging potential errors, style inconsistencies, and design-level issues, allowing human reviewers to focus on broader architectural or strategic concerns.
- *Project & Process Management*: GenAI Copilots are mentioned to enhance *Project & Process Management* by improving productivity [39, 47, 87], development time [7], effort estimation, completeness and consistency checks, and knowledge management. For Effort Estimation [79], they can analyze diverse historical data sources to provide more accurate time and resource predictions, helping managers plan and allocate effectively. By identifying gaps, redundancies, or conflicting logic across artifacts, GenAI Copilots may be able to ensure completeness & consistency throughout the SDLC [48]. Finally, Keim et al. [48] envision GenAI Copilots to enhance knowledge management throughout the SDLC as they are able to extract, summarize, and deliver contextually relevant information from different artifacts and sources, such as code, documentation, and requirements, thus supporting more effective onboarding, collaboration, and knowledge sharing.
- *Human & Team Factors*: Another area influenced by the emergence of GenAI Copilots are *Human & Team Factors*. They may support the development, collaboration, and well-being of individuals within software engineering teams. For onboarding and training [79], copilots can reduce ramp-up time by answering technical questions, explaining project contexts, and guiding new developers through unfamiliar codebases. In the area of Skills Learning [76], GenAI tools provide on-demand, personalized learning by offering code explanations, best practice recommendations, and interactive coding suggestions that help developers improve their competencies as they work. Team Coordination [76] may benefit from automated task summaries, meeting notes, and synchronization of project status, making collaboration smoother and more transparent. Communication-wise [47], GenAI Copilots can facilitate clearer and more efficient interactions between team members and/or stakeholders by, for example, translating technical concepts into simpler language, drafting documentation, or generating summaries for cross-functional stakeholders. Finally, they may also positively impact mental health [76] by reducing repetitive tasks or providing more structured and manageable workflows.

Across the reviewed literature, enhancements represent the most dominant impact of GenAI Copilots. Hou et al. [39] provide quantitative evidence, with most LLM applications enhancing mid-phase SDLC tasks: coding (57%), maintenance (23%), and quality assurance (15%). Wang et al.[92] echo this in the domain of software testing, with LLMs improving test generation, bug detection, program repair, and debugging, leading to faster testing cycles and increased coverage.

In summary, GenAI Copilots are reported and/or envisioned to enhance the SDLC by improving efficiency, quality, and collaboration, by enabling automation of tasks such as requirements engineering, coding, testing,

documentation, and project management, while also supporting human factors like learning, communication, and well-being.

4.2.2 *Reverses.*

- *Trustworthiness & Reliability* [34, 39, 48, 62, 64, 79, 92]: Though intended to streamline tasks, a major concern arises from "hallucinations" or plausible-but-incorrect code suggestions as well as misaligned designs. This lowers trustworthiness in generated artifacts. Moreover, inexperienced developers may accept erroneous suggestions without critical review, introducing errors and subtle bugs that compromise reliability.
- *Explainability* [34, 48]: Traditionally, explainability in software development flows from code to developer, meaning that because humans write the code, they inherently understand its logic, structure, and intent. With GenAI Copilots, this paradigm is reversed, and the developer must then interpret and validate the GenAI Copilot's output. This shift challenges explainability, as the rationale behind a copilot's suggestion is not always transparent or grounded in an easily traceable design decision.
- *Fairness* [2]: GenAI Copilots may generate code or logic that reflects patterns from biased training data, unintentionally encoding unfair assumptions or discriminatory behavior. Instead of proactively designing for fairness, developers may find themselves reacting to unintentional bias introduced by the copilot, scrutinizing and retroactively correcting outputs whose fairness implications are opaque.
- *Compliance* [64]: Instead of designing for compliance, developers may unknowingly introduce non-compliant constructs created by the AI. As a result, compliance becomes reactive, and developers must verify and audit AI-generated code post hoc, increasing the burden of compliance validation and introducing new risks if oversight is insufficient.
- *Sustainability* [64]: Traditionally, sustainability in software development (e.g., energy-efficient code, minimizing computational overhead, designing maintainable systems) is an intentional design goal. With GenAI-powered GenAI Copilots, this mindset can be inadvertently reversed: copilots often prioritize functionality and speed of development over resource efficiency or long-term maintainability. They may generate verbose, redundant, or suboptimal code that increases energy consumption or contributes to technical debt, especially if developers accept suggestions without critical evaluation.
- *Accountability* [64]: Developers and teams are usually directly responsible for the code they write and the decisions they make. With GenAI copilots, this clarity is blurred, effectively reversing accountability dynamics. As copilots generate code and design suggestions, it becomes more difficult to trace the origin of specific decisions or defects, especially when developers treat AI outputs as trustworthy by default. This creates ambiguity around who is responsible when something goes wrong: the developer, the organization, or the AI provider.
- *Individual Understanding Capabilities* [7, 87]: GenAI Copilots may reverse the individual understanding capabilities of developers, by encouraging overreliance or abstraction away from low-level detail.
- *Strict boundaries between SDLC stages* [84]: Strict boundaries between SDLC stages begin to dissolve, creating blurred transitions where coding, testing, and releasing collapse into a continuous loop. While this can enhance agility, it may also reverse long-established process controls, thereby increasing risks to traceability, defect localization, and process accountability.
- *Clear code ownership* [62, 64, 79]: Copilots produce code without clear attribution or traceable authorship, raising questions of accountability and provenance. Developers risk becoming curators rather than creators, which can weaken their sense of ownership and design rationale. Psychological ownership may likewise decrease – an effect observed for AI-generated text [18] – with yet unknown consequences for developer productivity and long-term maintenance.

- *Security* [39, 62]: With GenAI, security often becomes an afterthought, as developers may unknowingly incorporate insecure code patterns suggested by the AI, which is trained on a mixture of secure and insecure code.

Taken together, these reversed effects highlight the importance of critical oversight and contextual awareness in the deployment of AI within software engineering, to prevent inadvertently undermining productivity rather than enhancing it.

4.2.3 Retrieves.

- *Formal Requirements Specification* [26]: Ferrari and Spoletini argue that GenAI Copilots can retrieve the value of formal requirements specifications by lowering the barrier to entry. By automating or assisting in transferring classical natural language requirements to formal requirements, even developers or stakeholders without deep expertise in formal methods can retrieve those more rigorous and precise software specifications.
- *Formal Verification* [26]: Similarly, GenAI Copilots can act as a bridge between informal descriptions and formal verification logic. By generating or refining logical assertions, pre- and postconditions, and invariants, they help retrieve the rigor and guarantees offered by formal verification practices, which were once restricted to high-assurance systems and required highly specialized experts.
- *Rapid Prototyping* [79, 89]: Through their coding capabilities GenAI Copilots enable fast, exploratory implementations and, thus, can retrieve rapid prototyping where it was, beforehand, not feasible to perform.
- *Natural Language Explanation of Code* [87]: Storey et al. note that the widespread use of LLMs retrieves natural language explanations within code, a practice that echoes literate programming traditions. GenAI Copilots may generate intuitive summaries and explanations, making codebases more accessible and human-centric once again.

Together, these capabilities illustrate that GenAI Copilots do not simply retrieve practices, but rather help re-integrate enduring principles of software engineering that have long remained valuable yet were often too difficult or time-consuming to apply widely.

4.2.4 Obsolesces.

- *Manual debugging and bug reproduction* [76, 92]: Qiu et al. [76] and Wang et al. [92] highlight that test generation, bug reproduction, and oracle design are increasingly supported by GenAI Copilots, replacing manual processes.
- *Manual elicitation and formalization of requirements* [26, 89]: Ferrari and Spoletini [26] predict that the use of GenAI Copilots in requirements engineering and formalization may reduce the need for manual elicitation work, an idea echoed by Terragni et al. [89].
- *Need for Search of Code Snippets* [87]: Storey et al. assert that the need for searching code snippets is reduced or disappears by GenAI Copilots suggesting or even generating whole parts of the software.
- *Need for manually created documentation* [87]: They also discuss that the need for manually created documentation is reduced if GenAI Copilots are capable enough of generating informative and precise documentation.
- *Expert-only use of formal methods* [26]: Ferrari and Spoletini predict that the exclusive use of formal methods by experts may diminish, as GenAI Copilots can assist or even automate the extraction or generation of formal specifications from different sources, such as natural language requirements and code.

As a result of this analysis, we can state that each quadrant of the tetrad reflects deep shifts in software engineering due to the emergence of GenAI Copilots. Enhancements dominate current research, ranging from

automation of coding and testing to productivity and team dynamics. At the same time, developers and organizations must actively manage reversals, including diminishing understanding, blurred responsibility, and ethical concerns. GenAI Copilots retrieve formalism and creativity once undervalued in industry, while also rendering some traditional workflows and manual effort increasingly obsolete. This complex balance invites continuous investigation as the role of GenAI in the SDLC matures.

5 Impact of GenAI Teammates in SE

This section focuses on the GenAI Teammate form of GenAI augmentation and examines how (semi-)autonomous and goal-driven agents collaborate with human software engineers within development processes. As described in Section 3.1, GenAI Teammates differ from GenAI Copilots discussed in Section 4, as they are not merely invoked by humans but proactively participate in software engineering activities, sharing goals, responsibilities, and contextual awareness. This section presents the results of Cycle 2 of our methodology (see Section 2.2) and discusses the McLuhan tetrad derived from the collected evidence.

5.1 Rapid Literature Review

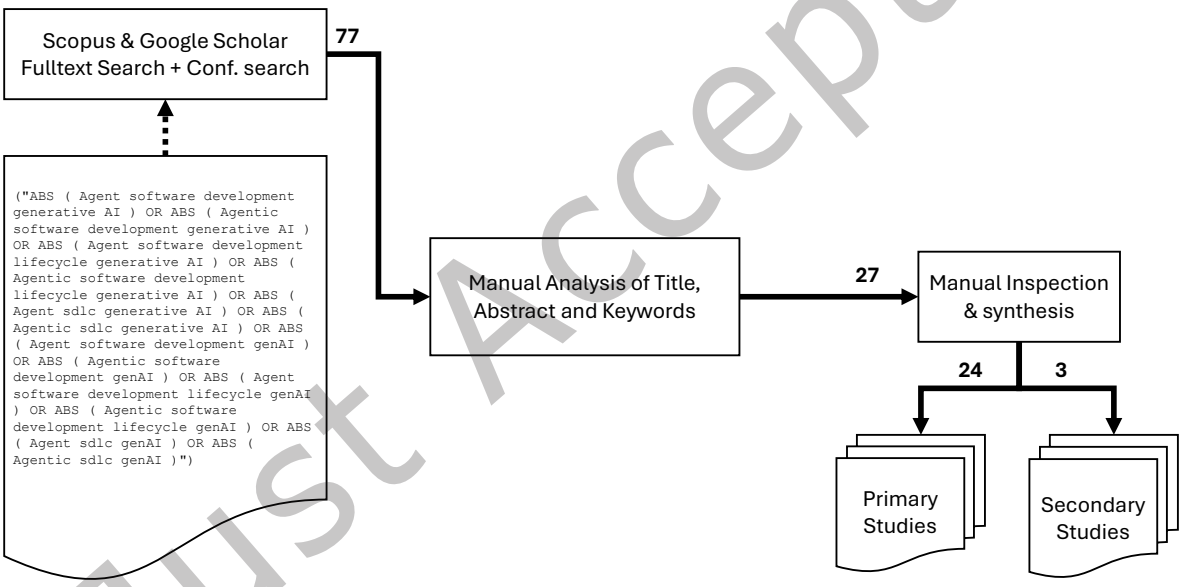

Fig. 5. Identifying relevant publications concerning GenAI Teammate

5.1.1 Review Approach. To include papers in the analysis of the perspective of GenAI Teammates, a mixed search approach – as a refinement of the one introduced in Section 2.2 – was conducted by searching among the papers accepted for the top-tier (A*) SE conferences (ICSE, FSE, and ASE), jointly with an effort-bound search on the academic database aggregator Google Scholar and Scopus. This refinement method was chosen to reduce the noise in the search by directly addressing the most important conference about the topic analyzed. The use of academic aggregators was nevertheless employed to complement focused research with extensive research on all

possible venues. Unlike the other RLRs, no AI support was utilized in the analysis of the papers because the total number of primary sources from the initial phase (77) was not sufficiently high to justify the use of automated assistance.

The process for conducting the rapid literature review is depicted in Figure 5. The search string is reported in Appendix A.2 and illustrated in Figure 5. The effort bound was set to 200 results; however, in both cases, this limit was never reached. The search was performed using the popular tool *Publish or Perish*[11] to facilitate a parameterized search on Google Scholar and Scopus. To include only the relevant sources in the non-peer-reviewed databases, the number of citations ranked the papers, and only those publicly accessible, written in English, with at least one citation were considered for inclusion in the initial pool of sources. Two inclusion criteria were set to ensure the inclusion of papers only relevant to the topic of Generative AI used as an Agent included in the SDLC: (i) the approach presented in the paper had to discuss multiple agents being able to act autonomously or on demand with additional capabilities beyond purely conversational ones (chatbots); (ii) the paper needed to explicitly discuss implication of the approach on the software development lifecycle.

Table 5. Relevant publications concerning GenAI Teammates

Paper	Authors	Year
ChatDev: Communicative Agents for Software Development [75]	Qian et al.	2023
Toward AI-facilitated Learning Cycle in Integration Course through Pair Programming with AI Agents [96]	Wei et al.	2024
SOEN-101: Code Generation by Emulating Software Process Models Using Large Language Model Agents [59]	Lin et al.	2024
From LLMs to LLM-based Agents for Software Engineering: A Survey of Current, Challenges and Future [44]	Jin et al.	2024
Experimenting with Multi-Agent Software Development: Towards a Unified Platform [82]	Sami et al.	2024
DevCoach: Supporting Students in Learning the Software Development Life Cycle at Scale with Generative Agents [93]	Wang et al.	2024
Autonomous Agents in Software Development: A Vision Paper [78]	Rasheed et al.	2024
CodePori: Large-Scale System for Autonomous Software Development Using Multi-Agent Technology [77]	Rasheed et al.	2024
Self-Evolving Multi-Agent Collaboration Networks for Software Development [40]	Hu et al.	2024
An Autonomous Multi-Agent LLM Framework for Agile Software Development [63]	Sanwal et al.	2024
COMMIT0: Library Generation from Scratch [106]	Zhao et al.	2024
An AI-native application assemble platform for easy-integrating of AIGC based services [45]	Jin et al.	2024
RepairAgent: An Autonomous, LLM-Based Agent for Program Repair [12]	Bouzenia et al.	2024
SALLMA: A Software Architecture for LLM-Based Multi-Agent Systems [10]	Becattini et al	2025
ProphetAgent: Automatically Synthesizing GUI Tests from Test Cases in Natural Language for Mobile Apps [51]	Kong et al.	2025
MUARF: Leveraging Multi-Agent Workflows for Automated Code Refactoring [103]	Xu et al.	2025
Knowledge-Based Multi-Agen [105]	Zhang et al.	2025
IDE Native, Foundation Model Based Agents for Software Refactoring [11]	Bellur et al.	2025
Facilitating Trustworthy Human-Agent Collaboration in LLM-based Multi-Agent System oriented Software Engineering [80]	Ronanki	2025
Evaluating Agent-based Program Repair at Google [81]	Rondon et al.	2025
Engineering LLM Powered Multi-Agent Framework for Autonomous CloudOps [72]	Parthasarathy et al.	2025
Developing Multi-Agent LLM Applications through Continuous Human-LLM Co-Programming [86]	Song et al.	2025
AGILECODER: Dynamic Collaborative Agents for Software Development based on Agile Methodology [69]	Nguyen et al.	2025
AI-Driven Automation in Agile Development: Multi-Agent LLMs for Software Engineering [49]	Khan et al.	2025
Multi-Agent Collaboration in AI: Enhancing Software Development with Autonomous LLMs [94]	Wasif et al.	2025
Multi-Agent Collaboration via Cross-Team Orchestration [19]	Du et al.	2025

[11] https://harzing.com/resources/publish-or-perish

5.1.2 Review Results. A total of 77 were identified via direct search to be considered for inclusion in the analysis; among these, 27 passed the screening phase by applying duplicate removal, inclusion, and exclusion criteria (details listed in Table 5). Among these, three secondary studies were found, and they were used to complement the information synthesized by directly analyzing primary sources.

5.2 McLuhan's Tetrad

To summarize the state of the art and the existing challenges in the domain of GenAI Teammates on the SDLC, the corresponding McLuhan's tetrad shown in Figure 6 was built to depict the latest advances in the field and identify future directions.

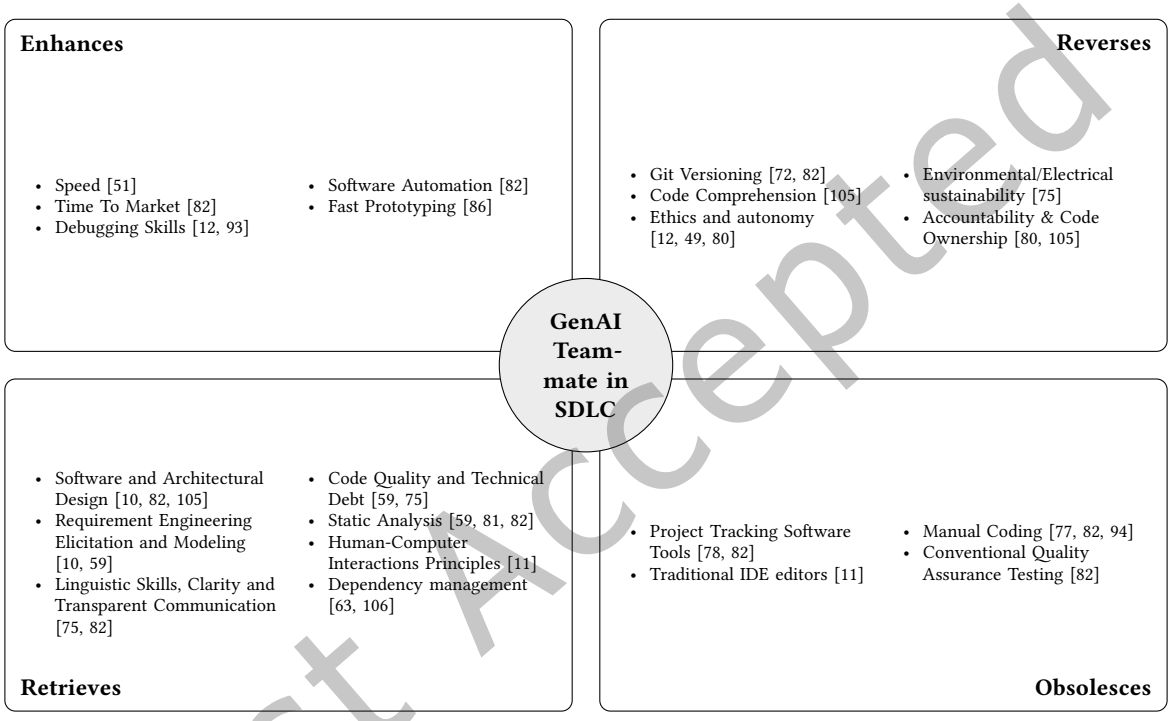

Fig. 6. McLuhan's tetrad derived from the rapid literature review of GenAI Teammate publications.

5.2.1 Enhances.

- *Software Automation*: Most research effort in Agent-based GenAI aimed a fully automating the software development process, relying on the agents to create software artifacts and descriptive documentation [82]. Some human-in-the-loop proposals emerged, aiming to put human developers, or more generally software engineers, in control of the whole process, characterizing the person with the ability to act as a validator of the generated software [80]. Other architectural proposals place the LLM agent(s) in the role of the verifier, double checking the solutions proposed by the model [45, 81].
- *Speed*: These agentic methodologies, relying on full or semi-automation, facilitate acceleration of the SDLC, improving developer efficiency [51].
- *Fast Prototyping*: The agentic approach can achieve rapid prototyping, allowing the developer to evaluate their artifact in the early stages of the SDLC [86].

- *Time To Market*: LLM generation capabilities, parallelized in an agentic approach, allow developers to quickly produce high-quality code and even entire programs, significantly expediting the SDLC, reducing the development effort, and accelerating project timelines [82].
- *Debugging Skills*: Improvements can be made in debugging competencies, which can gain from the expertise of specialized agents in the role of pair programmers, investigating the root causes of errors by elucidating error messages, thereby increasing the effectiveness of debugging [12, 93].

5.2.2 *Reverses.*

- *Git Versioning*: The commonly used Git Version control in a fully automated agentic approach may lack any remaining meaning [72]. Its original purpose of tracking versions and facilitating parallel programming human intent in assembling software may be reversed in an environment where every modification is automatically done by an agent, especially because most architectures do not parallelize code generation (mocking human teams' structure), but rather streamline the development pipeline with specialized agents [82].
- *Environmental Sustainability*: Heavy reliance on agentic architecture involves different LLM instances working jointly towards a common solution. Among the costs of this autonomy, an important communication overhead has to be taken into account, directly associated with potentially high energy costs, leading to environmental and sustainability concerns [75].
- *Accountability & Code Ownership*: Accountability challenges are partially addressed by frameworks that incorporate human oversight, holding individuals responsible for the software developed under their guidance. Specifically, responsibility assignment guidelines mandate that when an LLM-based multi-agent (LMA) system is assigned 'Responsible' status for a task, there must be at least one human actor assigned the 'Accountable' responsibility, thereby ensuring human control and validation over critical software development activities. However, in proposals for entirely autonomous systems, accountability continues to be an unresolved matter [80, 105].
- *Code Comprehension*: This landscape also includes situations in which individuals progressively diminish their proficiency in essential programming skills and code comprehension. Extensive codebases that were once developed through profound human expertise may now be constructed merely by depending on requirements definition: this could result in an increasing reliance on the tool, thereby losing control over the knowledge that underpins it. Furthermore, automated unsupervised solutions may prioritize functionality over readability, ultimately impairing the overall capacity for comprehension [105].
- *Ethics and Autonomy*: A core ethical mandate is respecting Human Autonomy and ensuring Human Agency and Oversight in AI systems. This is particularly challenging because agentic systems are designed to autonomously plan and execute actions to achieve goals, invoking tools and making decisions without human initiation for every step, mimicking a human developer's workflow [12]. This level of autonomy necessitates structured governance to maintain control, leading to frameworks like the RACI matrix, which explicitly mandates that when an LLM-based multi-agent (LMA) system is assigned the 'Responsible' status for a task, at least one human actor must be assigned 'Accountable' responsibility [80]. This mechanism ensures that human control is maintained over critical activities, aligning with trustworthy AI guidelines. Additionally, multi-agent systems, particularly those trained on large-scale code repositories, risk reinforcing existing biases in software development practices, which necessitate developing bias-mitigation techniques to ensure responsible AI deployment. Multi-agent systems may also pursue conflicting goals that should be taken into account when planning the overall system behavior. Ultimately, achieving trustworthy human-agent collaboration requires explicitly defining roles, ensuring human validation, and addressing the inherent lack of transparency in autonomous decision-making processes [49].

5.2.3 Retrieves.

- *Software Design*: Architectural definition becomes crucial both concerning the definition of the agents' architecture to be used, and the software architecture to be implemented [82, 105]. RAG mechanisms are also employed to retrieve knowledge from existing software designs (both from literature and from existing industrial documents), thereby empowering the agents' knowledge related to architectural patterns and design trade-offs, which influences their decision-making ability [10].
- *Requirement Engineering Elicitation and Modeling*: Multi-agent systems utilize roles such as the Requirement Engineer, Analyst, and Product Manager to replicate human development activities. These specialized agents actively apply knowledge by interpreting natural language user input and translating it into structured artifacts, such as precise user stories containing clear acceptance criteria [10, 59]. To make the most of the agents' skills, an effective requirements elicitation
- *Linguistic Skills, Clarity, and Transparent Communication*: Other non-technical skills, such as linguistic ability and unambiguity, enable a clear and transparent specification of the software to be implemented. Clarity is the key aspect in all the input artifacts provided to any agentic approach to obtain a working implementation consistent with the end-user needs [75]. To enforce clarity and precision across the whole process, agents leverage specialized Prompt Engineering techniques and employ structured mechanisms like communicative dehallucination, which systematically requires the assistant agent to proactively seek more detailed suggestions from the instructor agent before providing a formal solution [82].
  Although the Waterfall model is often applied in systems requiring a linear path, the agentic emulation demonstrates how these established software process models can enhance the quality and stability of LLM-generated code by organizing collaborative activities [59].
- *Code Quality and Technical Debt*: Given the probabilistic nature of the implementation provided by the agents, assessing software quality becomes essential to minimize code debt [75]. Metric-driven assessments, refactoring practices, and code review should be used to maintain an acceptable code quality. Furthermore, fine-grained analysis of how development activities impact code quality reveals that a dedicated phase like code review is crucial for reducing the density of code smells and improving code reliability [59].
- *Static Analysis*: Static quality analysis and functional validation become essential steps in creating robust code. Automatic or manual validation of AI-generated code becomes essential to avoid pitfalls [59, 81]. Specialized agents can leverage external tools to detect and categorize code smells, providing objective feedback that guides code refinement. By integrating these specialized quality checks and iterative refinement loops, the agentic approach systematically combats the creation of low-quality, high-technical-debt code, bringing existing software quality knowledge directly into the autonomous execution cycle [82].
- *Human-Computer Interactions Principles*: As the automated generation of software through agentic AI has demonstrated the ability to create functional solutions with rather naive or no graphical interfaces, a critical challenge that must be tackled by leveraging the existing body of knowledge is the effective design of interactions between humans and technology, drawing upon principles of Human-Computer Interaction [11].
- *Dependency Management*: Although some agentic approaches can autonomously handle broader project management tasks related to dependencies, such as migrating a dependency management system, updating dependency files, resolving compatibility issues, and generating necessary lock files [63], in a completely automated setting, the challenge of managing dependencies and ensuring that libraries are up to date is a significant hurdle that requires appropriate expertise to navigate [106].

5.2.4 Obsolesces.

- *Manual Coding*: The obsolescence of manual coding is cemented by the agentic AI's ability to handle iterative development, debugging, and quality assurance autonomously [94]. Agents autonomously

generate executable code segments and iteratively refine them through collaborative exchanges and internal feedback loops. Since agentic AI effectively automates complex and time-consuming technical tasks, the role of human software engineers shifts from manual coders to that of oversight and strategic decision makers, allowing them to focus on more complex and innovative problem solving activities [77, 82].

- *Project Tracking Software Tools*: Autonomous agentic environments aim to seamlessly link all phases of development, incorporating a continuous feedback loop where discussions among stakeholders are automatically transformed into actionable items, such as prioritizing features in the project's backlog, generating user stories, and suggesting test cases, all without manual intervention. This integrated and automated approach suggests a paradigm shift in which task and project management artifacts are generated and maintained fluidly within the AI development ecosystem itself, reducing the dependence on separate dedicated project tracking software [78, 82].
- *Traditional IDE editors*: Although some research emphasizes that IDEs remain the "ideal place" for agents to reside because they provide critical safeguards and static analysis APIs that can be utilized as tools by the agent, the agent's ability to automate core code generation, editing, and quality assurance processes inherently reduces the constant necessity for manual file opening and editing within the visual editor interface, ultimately yielding to a chat-oriented development environment [11].
- *Conventional Quality Assurance Testing*: Agentic AI significantly obsolesces traditional and conventional Quality Assurance (QA) testing by integrating automated testing and verification directly into autonomous software development [82]. Multi-agent systems, leveraging specialized agents like Developers and Testers, streamline the QA process by autonomously generating and executing comprehensive test suites, thereby reducing the need for continuous manual testing efforts. This autonomous generation and execution of diverse tests, from unit and end-to-end tests to GUI and robustness checks, integrates QA as a continuous, intrinsic element of the multi-agent workflow, rather than a separate, manual phase.
  Human involvement gains a pivotal role in clearly expressing desired goals and tasks for the GenAI Teammate, currently in the form of prompts. This prompt-based programming obsolesces traditional Integrated Development Environments and their built-in features, such as conventional debugging and testing strategies. Rethinking Quality Assurance and Program Correctness strategy becomes crucial to deliver code without the burden of technical debt [82].
  Agentic Gen-AI research is clearly centered around the automatic software generation, directly from specifications. In this scenario, the creation of "off-the-shelf" solutions can largely lose its appeal. This can happen since adapting existing software in a specific process is more complex compared to automatically generating software tailored to users' and companies' needs. The same phenomenon can cause the obsolescence of project tracking software tools, which will likely turn into requirements definition management software to gather user needs and organize requirements to be dispatched to the agentic architecture for software evolution [93].

6 Impact of GenAIware in SE

This section reports the results of Cycle 2 of our methodology (see Section 2.2) for the GenAIware form of GenAI augmentation. The corresponding literature review investigated how GenAIware manifests in software systems whose functionality is partially or fully realized through GenAI components, embedding intelligence directly into the tools and environments that support software engineering activities.

6.1 Rapid Literature Review

6.1.1 Review Approach. For the literature review, we followed the process depicted in Figure 7. We report further details on the search strings and the prompts to filter the results for interested readers in Sections A.3 and B.3.

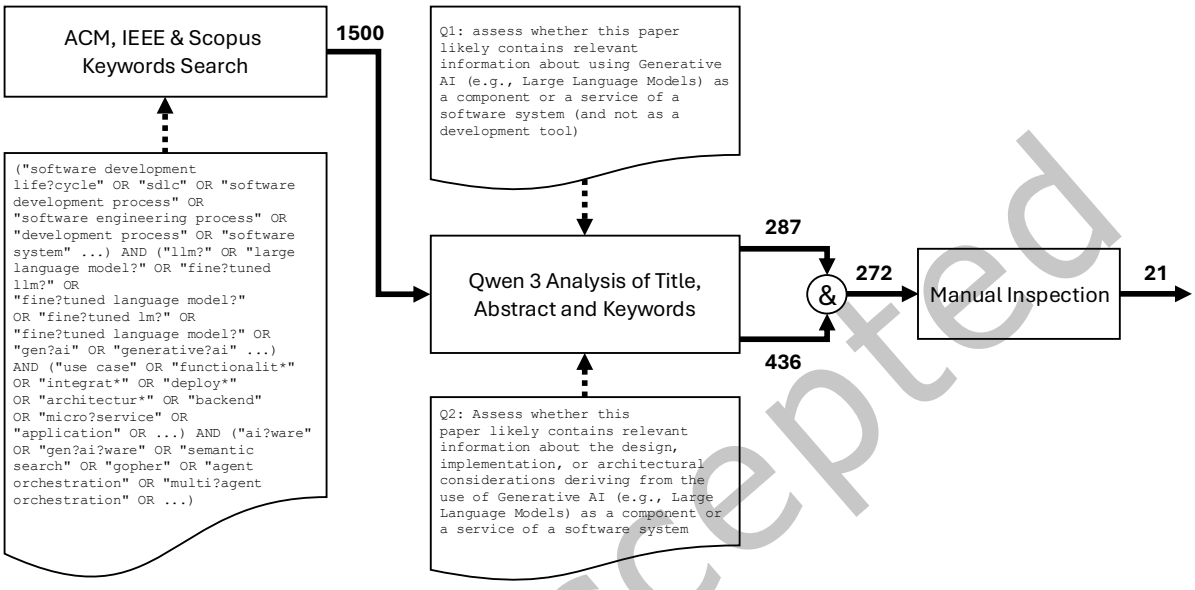

Fig. 7. Identifying relevant publications concerning GenAIware

Due to the topic of GenAIware has only recently gained traction, we slightly adapted the process from Section 2.2.1. We started the rapid literature review searching results on full-text and metadata from ACM and IEEE, and on metadata only on Scopus. We focused on papers published since January 2019, following the release of the main Transformer-based language models, namely GPT and BERT. We did not limit ourselves to the main SE venues, because GenAIware is often only reflected in specific events dedicated to this topic, such as the ACM International Conference on AI-powered Software (AIware). Yet, we only considered the top 500 results from each source for a total of maximum 1500 papers.

To find GenAIware-related work, we adapted the search terms to reflect, besides SDLC and GenAI (i) software components and processes (e.g., "deployment", "architecture"), and (ii) GenAIware-specific keywords (e.g., "prompt engineering", "ai-ware").

We automatically filtered the initial results using the Qwen3 LLM to help reduce the burden of manually assessing which of the papers were relevant to the topic. Given the title, abstract and keywords of each retrieved paper, we prompted the LLM with two questions, in as many separate contexts, covering the two aspects of GenAI characterizing GenAIware. Specifically, we asked the LLM these two questions: (Q1) assess whether the paper likely contains relevant information about using Generative AI (e.g., Large Language Models) as a passive element (e.g., as a component or a service of a software system), and (Q2) assess whether this paper likely contains relevant information about the design, implementation, or architectural considerations deriving from the use of Generative AI (e.g., Large Language Models) as a component or a service of a software system. Q1 was aimed at

highlighting papers where GenAI is augmenting the developed software system and Q2 was aimed at highlighting papers where GenAI was integrated as a component or a service of the software system. We retrieved the papers that received a positive answer to both questions, for a total of 272 papers.

We manually processed these LLM-selected papers by taking care of removing any remaining papers not relevant to the scope of the review. At this step, we noticed many false positives coming from papers concerning the integration of GenAI Copilot into software development tools. Additionally, we discarded all papers presenting or discussing application-specific solutions, which – despite the fact that they represent a valuable reference for the development of GenAIware – are outside the scope of this review. After this last manual filtering step, we were left with 21 papers.

6.1.2 Review Results. We report the main information about the papers retrieved from the rapid literature review in Table 6. The most covered topics were architectures for and integration of GenAI in a software system. This was followed by topics related to prompt engineering and information retrieval. Finally, the few remaining papers discuss requirements, security and safety of GenAI-augmented software.

From the architecture perspective, several patterns have been presented to exploit existing models depending on the target task, often with an agent-oriented perspective [61]. These patterns for architectures, mainly thought for LLM, prescribe how to manage memory (short- and log-term), schedule model queries and execution, possibly with accessory plugins (like guardrails). Frameworks like DAWN [4] or SALLMA [10] focus on distribution and multi-agent use cases. Finally, security is becoming a concern already at architectural-level when the software is backed by GenAI, especially when moving to enterprise-level solution [46].

The challenges concerning the integration are the lack of detailed interface specifications and the variability of the model's output [85], which often cause defects affecting functionality, efficiency and security [85]. As a result, traditional software engineering practices are exposed to new failure modes and the subsequent need for testing, hence requiring manual effort, which in turn leads to inherent subjectivity in evaluating LLM outputs [68]. Emerging solutions are pushing for clean component interfaces, as well as input preprocessing and output postprocessing [85]. On top of these issues, there are those related to the integration of the Machine Learning processes into the pipeline, which require accounting for data dependencies management and Machine Learning model lifecycle [50].

As mentioned before, memory plays a special role [61]. Long-term memory plays a central role in dealing with challenges given by hallucinations, outdated knowledge, and domain-specific gaps, thus the introduction of Retrieval Augmented Generation (RAG)-powered applications for information retrieval [8]. These applications enable better reliability and trustworthiness of LLM-powered software. Building these applications implies the choice of the appropriate data store (e.g., vector databases for semantic search, that is, replacing traditional lexical search) and the correct preprocessing of the stored documents [8]. User data can be stored and retrieved as well, introducing the need for privacy-related requirements.

Finally, we are witnessing the development of prompt programming, as prompts for LLMs are becoming a meticulously engineered part of the code-base, functioning as programs themselves [58]. What makes prompting challenging is their ever-evolving nature as they strongly depend on the LLM they are given to, which may change, and on the current knowledge about that LLM's capabilities, which is again changing through time. Only recently, approaches for rigorous prompt engineering [6] and prompt sharing [5, 25] are emerging, leaving otherwise the task subject to a naive trial and error approach [58].

Additionally, emerging topics concerning include requirements engineering, safety and security. Requirements are affected by the stochastic and probabilistic nature of Machine Learning models in general, leading to the need to define a "window of acceptable behavior" or "soft-goals" rather than traditional hard constraints [21]. These soft constraints are also applied in terms of the safety of the generated content, as challenges about the

Table 6. Relevant publications concerning GenAIware

Paper	Authors	Year	Covered topic(s)					
			Requirements	Architecture	Integration	Safety	Data	Prompt
Seven Failure Points When Engineering a Retrieval Augmented Generation System [8]	Barnett et al.	2024					✓	
CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural Language Programming [25]	Feng	2024						✓
Secure Software Architecture for Enterprise Generative Artificial Intelligence [46]	Joshi	2024		✓		✓		
Securing Applications of Large Language Models: A Shift-Left Approach [53]	Lan et al.	2024				✓		
A Taxonomy of Foundation Model based Systems through the Lens of Software Architecture [60]	Lu et al.	2024		✓				
Towards Responsible Generative AI: A Reference Architecture for Designing Foundation Model Based Agents [61]	Lu et al.	2024		✓			✓	✓
Feature Model-based Integration of Machine Learning in Software Product Lines [50]	Kholkar et al.	2024			✓			
Requirements Elicitation for Machine Learning Applications: A Research Preview [21]	Elvira at al.	2024	✓					
Tutorial on Landing Generative AI in Industrial Social and E-commerce Recsys [102]	Xu et al.	2025			✓			
Iterative Proof-Driven Development LLM Prompt [6]	Bakharia	2025						✓
Prompts Are Programs Too! Understanding How Developers Build Software Containing Prompts [58]	Liang et al	2025						✓
Supporting Students in Prototyping AI-backed Software with Hosted Prompt Template APIs [5]	Aveni et al.	2025						✓
SALLMA: A Software Architecture for LLM-Based Multi-Agent Systems [10]	Becattini et al.	2025		✓				
Beyond the Comfort Zone: Emerging Solutions to Overcome Challenges in Integrating LLMs into Software Products [68]	Nahar et al.	2025			✓			
Verification and Validation of LLM-RAG for Industrial Automation [66]	Min et al.	2025				✓		
An Architecture and Protocol for Decentralized Retrieval Augmented Generation [36]	Hecking et al.	2025		✓			✓	
A Functional Software Reference Architecture for LLM-Integrated Systems [13]	Bucaioni et al.	2025		✓	✓			
Are LLMs Correctly Integrated into Software Systems? [85]	Shao et al.	2025			✓			
Towards Retrieval-Augmented Large Language Models: Data Management and System Design [24]	Fan et al.	2025		✓			✓	
Development of AI Agent Based on Large Language Model Platforms [15]	Chen et al.	2025			✓			
DAWN: Designing Distributed Agents in a Worldwide Network [4]	Aminiranjbar et al.	2025		✓				

generation of toxic, untruthful or unfaithful [42] content do not allow for 100% solutions [46, 53]. Nevertheless, some use cases still require hard constraints which result in tailored Verification and Validation frameworks [66].

6.2 McLuhan's Tetrad

Our rapid literature review informed each quadrant of the tetrad as shown in Figure 8.

6.2.1 Enhances.

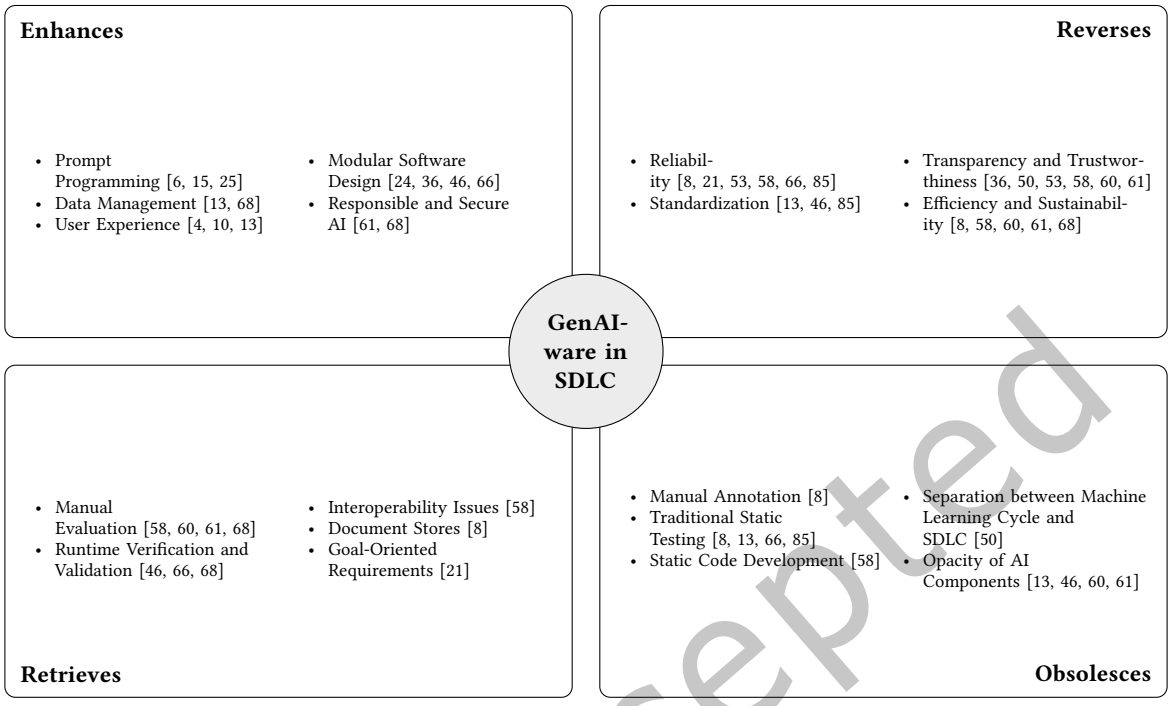

Fig. 8. McLuhan's tetrad derived from the rapid literature review of GenAIware publications.

- *Prompt Programming*: The introduction of GenAI has altered the programming paradigm, extending it to include dynamic prompts and data, along with code. Prompts allow for the programming of the behavior of LLMs, resulting in a form of natural language coding or prompt programming [58]. As a result, prompt engineering developed into an actual research area, providing structured approaches to create and (re-)use prompts and strategies to manage the context [6, 15, 25]. At the same time, data are necessary to prompt and evaluate these generative models. RAG and few-shot learning rely on input data to construct the prompt and manage the context as well.
- *Data Management*: Data management, especially information retrieval, has undergone a strong makeover [24, 36]. Semantic search approaches are now integrated into search engine architectures, pushing for the enhancement of vector databases to support the new indexing approach. LLMs have become the new interface to unstructured and semi-structured data stores and, at the same time, LLMs result in more reliable and trustworthy having access to updated and trusted data sources [46, 66].
- *User Experience*: In addition to search engines, the integration of AI assistants in existing applications enhances the user experience, providing an interactive interface that can search the application and respond in a human-understandable language [13, 68]. This experience is further improved by improved customization. In fact, user-specific or domain-specific data can be plugged into the generative model without needing any training or fine-tuning.
- *Unified Software Design*: On the design and architecture side, we are finally observing steps toward unified interfaces for these models [4, 13]. This unification has multiple positive effects. It helps the modularization of software architectures using these models, it enhances the automation of deployment and testing and it helps the interoperability with and among agentic approaches [4, 10].

- *Responsible and Secure AI*: The integration of AI modules in software systems and applications is raising awareness towards the risks connected to its use. As a result, we are observing interests in non-functional aspects like responsible AI and security [61, 68]. Security is not only limited to data privacy, but also to the safety of user inputs and generated content, leading to the development of dedicated testing approaches and guardrails [68].

6.2.2 *Reverses.*

- *Reliability*: GenAI models are de facto probabilistic tools; this nature clashes with the control, predictability, and replicability of traditional software. For example, LLMs are stochastic [8, 21, 66], which introduce new challenges for validating GenAI-augmented software [8, 85]. At the same time, the issue of hallucinations may compromise trust in GenAIware [8, 53, 66]. Moreover, prompts, being fragile, model-dependent, and context-dependent, often push the development toward rapid, unsystematic, trial-and-error practices rather than structured processes [58].
- *Standardization*: The existence of multiple, independent, and different interfaces to GenAI models challenges modularity, creating difficulties in system composition and management due to insufficient specifications and varied contextual requirements [13, 46, 85]. Several solutions, such as the MCP (Model Context Protocol) and A2A (Agent-to-Agent Protocol), have been proposed; however, the absence of a (yet) universally adopted framework enforcing consistent interface definitions and interoperability across platforms and vendors has resulted in fragmented ecosystems.
- *Transparency and Trustworthiness*: Quality attributes and the development process are also affected. The opaque nature of GenAI models obscures transparency and lack of control, complicating fault localization and scattering responsibility across multiple components [36, 50, 58, 60, 61]. Furthermore, biases arising from training data manifest in outputs that can be discriminatory or unfair, making ethics an ongoing challenge and a fixed requirement [53, 60, 61].
- *Efficiency and Sustainability*: From an operational standpoint, efficiency and sustainability are reversed in exchange for escalating costs and more difficult maintenance. Relying on external APIs may introduce latency and high costs, while local deployment of large GenAI models may require significant computational and human resources [8, 58, 68]. As these models evolve, their behavior drifts over time, demanding continuous calibration, monitoring, and adaptation. This dynamism disrupts established workflows and erodes long-term stability, reversing the traditional pursuit of cost-effectiveness and reliability in software systems [8, 58, 60, 61].

6.2.3 *Retrieves.*

- *Manual evaluation*: Development once again becomes grounded in human-intensive, iterative practices, with prompt engineering emerging as a central trial-and-error activity where prompts are often crafted and revised to enforce the desired agent behavior [58, 60, 61]. Much of the manual effort that was thought to be lost resurfaces in new forms, particularly in data annotation and curation, which become essential for building domain-specific, high-quality datasets [58, 68]. Evaluation and testing similarly shifts back towards more subjective, qualitative inspection, as the adequacy of generated outputs can be often evaluated only through human qualitative analysis [58].
- *Runtime Verification and Validation*: Architecturally, GenAIware retrieves and reshapes persistent software engineering challenges. The dynamic and probabilistic nature of GenAI revives the need for continuous runtime evaluation and adaptive verification and validation frameworks [46, 66, 68].
- *Interoperability Issues*: The challenges of integrating opaque machine learning elements return emphasis on transparency [58].

- *Document Stores*: When using RAG (retrieval augmented generation), long-standing document stores and search infrastructures regain prominence, now replaced by vector databases that enable semantic search and provide external, trusted, and verifiable knowledge to complement internal GenAI model memories [8]. In this way, GenAIware not only revives but also reconfigures older practices and concerns of information retrieval systems.
- *Goal-Oriented Requirements*: Concerning requirements engineering, rigid and deterministic specifications no longer align with the stochastic and unpredictable behavior of LLMs. Goal-oriented requirements engineering approaches, which focus on soft goals and acceptable behavior ranges, could be adapted to address the inherent uncertainty and flexibility required in the development of the ML system [21].

6.2.4 Obsolesces.

- *Manual Annotation*: In information retrieval, the rise of RAG reduces the need for labor-intensive manual annotation, further eroding the role of static methods. This last effect extends to many cases that require human-in-the-loop approaches [8].
- *Traditional Static Testing*: In terms of quality assurance, deterministic test-driven development is challenged by the probabilistic and uncertain outputs of GenAIware, whose evaluation goes beyond a simple binary pass/fail metric [66]. Moreover, static, quantitative evaluation techniques also lose relevance, as prompt programming and RAG systems demand runtime, qualitative, and operational assessment as the underlying GenAI model and data change [8]. Advances in vector databases reinforce this shift, making traditional storage, indexing, and retrieval mechanisms less capable of supporting rapid changes in models and data [13, 85].
- *Static Code Development*: At the process and organizational level, the experimental nature of prompt programming makes the prompt engineering stage detached from the slow and rigid code review cycles, which turn out less suitable for practical development [58].
- *Separation between Machine Learning Cycle and SDLC*: The long-standing separation of machine learning development from the broader SDLC is becoming obsolete as integration of AI-based elements becomes relevant across all stages, from requirements to deployment [50].
- *Opacity of AI elements* the demand for responsible AI, transparency, and explainability makes purely opaque integration unsustainable, replacing it with approaches that emphasize explainers, guardrails, and continuous monitoring to ensure trustworthiness and accountability [13, 46, 60, 61].

7 Impact of GenAI Robots in SE

This section presents the results of Cycle 2 of our methodology (see Section 2.2) concerning the GenAI Robot form of GenAI augmentation. The corresponding literature review examines how (semi-)autonomous and goal-driven agents deliver part of the functionality of software systems or applications. As discussed in Section 3.1, GenAI Robots differ from GenAI Teammates in that they are not involved in the development process itself but become integral elements of the resulting software product, operating as self-contained AI components once deployed.

7.1 Rapid Literature Review

Given the wide range of application domains in which software systems can be realized to deliver parts of their functionality through GenAI Robots, we opt for a broad search and review, as explained in the following.

7.1.1 Review Approach. The rapid literature review approach we performed for GenAI Robots is depicted in Fig. 9.

We started with a keyword search using Elsevier's Scopus database (note that Scopus, since 2021, also includes arXiv preprints). The keyword search gave us fine-grained control over the search terms (providing synonyms

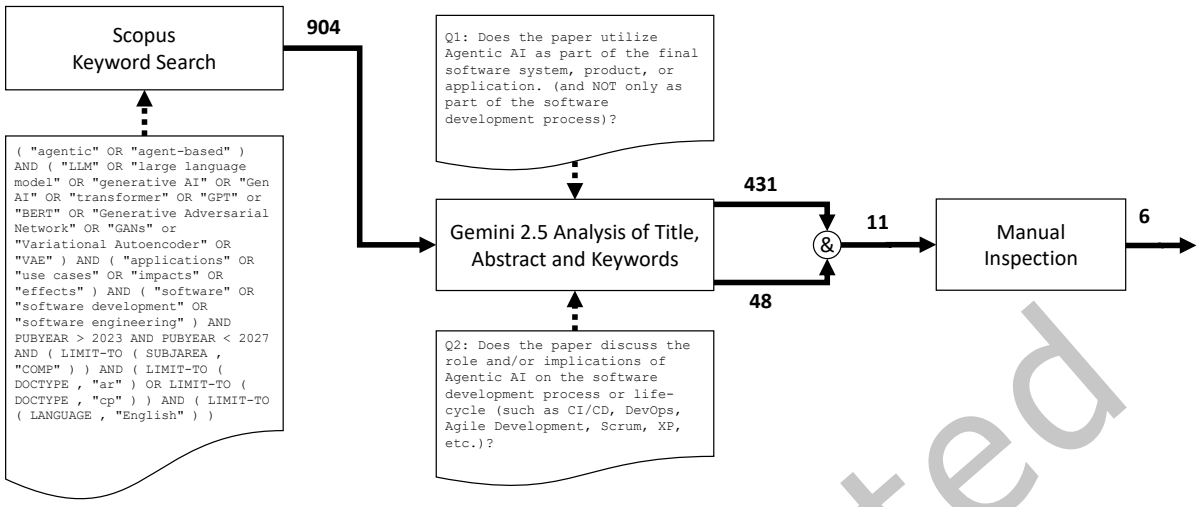

Fig. 9. Identifying relevant publications concerning GenAI Robots

for agentic AI and requiring GenAI as a technique), while also being able to explicitly state inclusion criteria (publications from 2024 and beyond) and exclusion criteria (only articles and conference publications in English). This Scopus search resulted in 903 publications.

As a second step, we then employed Google's Gemini 2.5 Flash via its programmatic API to aid us in the analysis process. We prompted Gemini to scrutinize the title, abstract and keywords to answer two specific questions: (Q1) whether the paper actually covers the use of GenAI as a Robot (and not as a GenAI Teammate, for instance), and (Q2) whether the paper explicitly discusses impacts or implications for the SDLC. For each of these questions, we not only asked for a yes/no answer but also required Gemini to provide its rationale for giving the respective answer. The full prompt is provided in Appendix B.4. This resulted in 11 publications for which both answers were yes.

Finally, we manually inspected these 11 publications and eliminated the ones that were out of scope, resulting in 6 relevant publications.

7.1.2 Review Results. The relevant publications on GenAI Robots are listed in Table 7.

We observe that given the very low number of relevant publications, GenAI Robots are clearly an emerging area. Even though, GenAI Robots were used in several different application domains. Two of the papers focus on individual tasks, i.e., on isolated aspects of the lifecycle. For example, Mishra et al. focus on the task of testing retail systems. Three of the papers address broader lifecycle aspects by introducing domain-specific development processes. For example, Lee et al. integrate the use of GenAI Robots into the Double Diamond lifecycle from (industrial) design[12]. However, these papers lack alignment and integration with an overall software development lifecycle.

7.2 McLuhan's Tetrad

As seen above, there is only little existing work on how GenAI Robots may impact SE, indicating that this is a nascent research area. What this means is that the below McLuhan's tetrad and roadmap items are necessarily more speculative than those for the other, more established forms. To accommodate the fewer existing work, we

[12]https://www.designcouncil.org.uk/our-resources/the-double-diamond/

Table 7. Relevant publications concerning GenAI Robots

Paper	Authors	Year	Application Domain	SDLC Aspects
A Conversational Assistant Framework for Automation [20]	Duesterwald et al.	2024	BPM	Overall: domain-specific lifecycle
OMACS Based Adaptive Intelligent Tutoring System Development [31]	Gowri	2025	Education	Overall: domain-specific lifecycle
When and How to Use AI in the Design Process? Implications for Human-AI Design Collaboration [54]	Lee et al.	2025	(Industrial) Design	Overall: domain-specific lifecycle
Retail Resilience Engine: An Agentic AI Framework for Building Reliable Retail Systems With Test-Driven Development Approach [67]	Mishra et al.	2025	Retail	Single Task: testing
Business Compliance Detection of Smart Contracts in Electricity and Carbon Trading Scenarios [101]	Wu et al.	2024	Electricity	Single Task: compliance verification

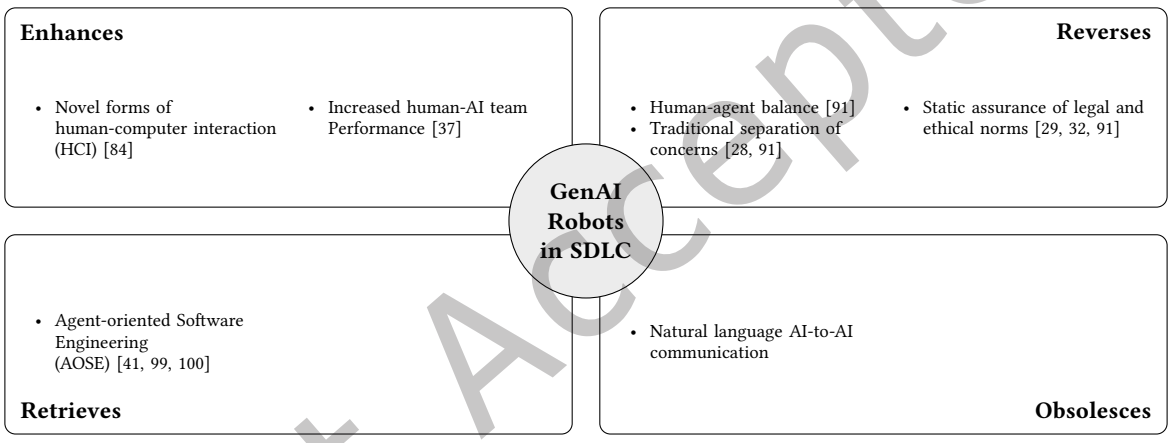

Fig. 10. McLuhan's tetrad for GenAI Robots.

also explore more general challenges identified for GenAI Robot in the literature, leading to the following items of the tetrad, also shown in Figure 10.

7.2.1 Enhances.

- *Novel forms of human-computer interaction (HCI):* Having agents – in the form of GenAI Robots – deliver parts of the software systems' functionality opens up novel avenues for HCI. Instead of humans initiating the interaction with the "computer" (i.e., software), it can now be the GenAI Robots that initiate this interaction. This leads to several different forms of human-agent interactions that may be chosen to realize a software application. As an example, Sauvola et al. describe four different scenarios with increasing AI participation [84]. As another example, Stanford's Social and Language Technologies Lab (SALT) envisions five levels of such interaction, ranging from fully human to fully AI task completion[13]. Altogether, this

[13]https://futureofwork.saltlab.stanford.edu/

raises the question of where in the SDLC such a decision is taken, or whether there is a need to adapt and evolve the initial decision.

- *Increased human-AI team performance:* Combining GenAI Robots and humans in a complementary manner offers an increase in overall team performance, i.e., a level of performance that neither of them can individually achieve [37]. Or, in other words, we see the emergence of hybrid intelligence, where humans and GenAI complement each other rather than compete. Of course, this raises the question of when and how to determine what such a human-AI team should look like.

7.2.2 Reverses.

- *Human-agent balance:* One key concern is how best to leverage AI capabilities and human expertise, thus increasing efficiency while maintaining human judgment and creativity [91]. This requires defining clear roles and responsibilities between humans and GenAI Robots, including identifying situations where human intervention is necessary; for example, in the case of failure of GenAI Robot s [91]. In general, human-agent teaming raises questions to be answered during the SDLC, such as whether, for a given task, human actors will collaborate with agents as peers or whether humans take on supervisory roles with selective intervention.
- *Traditional separation of concerns:* The SDLC has to provide activities to determine which parts of the functionality of the application software will be carried out by GenAI Robots and which parts of manual human tasks can be automated by GenAI Robots. This may go significantly beyond the traditional SE concept of separation of concerns, such as object orientation, and may require a much deeper analysis. On the one hand, it may require analyzing historical data of how previous applications were used [28]. On the other hand, it may require clear metrics and performance indicators such that the impact of GenAI Robots on process efficiency (e.g., in terms of cost and time, decision quality, and broader organizational outcomes) can be measured at run time [91].
- *Static assurance of legal and ethical norms:* It must be ensured that GenAI Robots comply with legal norms, while considering the context and environment in which they operate [91]. Given the greater degree of autonomy of GenAI Robots, using static rules to ensure compliance with these norms may no longer be feasible, making GenAI Robot s find unexpected and potentially harmful ways to achieve their goals [29]. We thus require ”guardrails”, which codify norms in such a way as to ensure compliance also in unspecified, unstructured situations [32]. As abstractions for managing such guardrails, the notion of (normative) frames may be explored. In contrast to more operational notions of declarative or procedural abstractions, frames rely on deontic logic, which specifies obligations, permissions, and prohibitions[14]. The definition of guardrails at design time should be complemented by safety measures that monitor GenAI Robot performance at run time, thus helping to observe and prevent unwanted adaptations or evolutions of GenAI Robots [91].

7.2.3 Retrieves.

- *Agent-oriented Software Engineering (AOSE):* Now that powerful GenAI makes the vision of agent-based systems realistic, it is worth revisiting SE research on building agent-based systems, which was conceived more than two decades ago [41, 100]. The principles, methods and processes of agent-oriented software engineering (AOSE) provide a strong foundation for novel approaches to developing, operating, and maintaining GenAI Robots. An example is the Gaia methodology [99], intended to allow a software engineer to go systematically from a statement of requirements to a design that is sufficiently detailed for it to be implemented. Gaia provides an agent-specific set of concepts through which a software engineer

[14]http://www.dagstuhl.de/25192

can understand and model a complex system, such as responsibilities, activities, interaction protocols, and permissions.

7.2.4 Obsolesces.

- *Natural language AI-to-AI communication:* If GenAI Robots communicate with other GenAI Robots or traditional software, it is no longer necessary for them to converse in natural language. Other forms of communication may become much more effective. As an example, GenAI Robots realized via LLM may directly exchange the tokens and not the original natural-language input. As a more extreme example, interacting GenAI Robots may even come up with their own efficient language, as demonstrated by the Gibberlink project[15]. This raises the question when and how in the SDLC to balance efficiency with transparency, i.e., the capability for human understanding of the AI-to-AI language.

8 Research Roadmap

Resulting as an outcome of Cycle 3 of our methodology (see Section 2.1), this section synthesizes an overall research roadmap on GenAI augmentation in SE. To do this, we analyze the four McLuhan tetrads and identify transversal patterns, recurring themes, and complementary insights. The roadmap includes the four forms of GenAI augmentation in SE with the associated challenges and opportunities. The mapping of these challenges and opportunities to their sources (i.e., the analysis via the McLuhan's tetrad) offers a valuable lens for understanding why certain phenomena emerge and how their side effects can be anticipated, for instance, how an innovation that enhances a practice may simultaneously reverse or obscure critical aspects, creating new needs for interoperability and compliance. This dual perspective, operational and theoretical, supports a multidisciplinary research agenda.

The overall roadmap consists of five parts, Parts A–D (Sections 8.1–8.4) for each form of GenAI augmentation, and Part X (Section 8.5) for cross-form aspects.

8.1 Roadmap Part A: GenAI Copilots

Table 8 shows the challenges and opportunities in augmenting the process with GenAI Copilots. The table systematizes key research challenges in the integration of GenAI into software development workflows, along with corresponding opportunities for innovation. The challenges span both technical and social dimensions, ranging from the specification of reliable interaction interfaces (A1) and mechanisms for traceability and accountability (A2) to quality assurance of generated artifacts (A3) and the long-term maintenance of evolving AI elements (A4). Several entries highlight the need to embed GenAI Copilots seamlessly into established development processes (A5), while maintaining developer agency and transparency (A6). Broader concerns include addressing bias and fairness in domain-specific contexts (A7) and enabling collaborative work across multiple stakeholders (A8). Finally, systemic risks are considered, such as dependency management, safety, and security (A9), as well as the fundamental question of how to establish trustworthiness in GenAI-assisted development (A10). The outlined opportunities point to a research agenda encompassing formal abstractions, hybrid verification techniques, orchestration mechanisms, debiasing methods, lightweight inference strategies, and governance frameworks, all aimed at creating reliable, sustainable, and accountable GenAI Copilots for software engineering.

Table 9 maps the research challenges and opportunities to the source that triggered the form of GenAI Copilot augmentation. The source can be one or more elements in the McLuhan's tetrad. A clear and pragmatic research framework naturally emerges from the mapping. The introduction of GenAI elements as copilots in software development processes brings both significant technical opportunities (formal prompt abstractions, provenance tracking, hybrid verification, CI/AI pipelines, orchestration tools) and systemic and social risks (opacity, bias, dependency, degradation over time, and collaborative conflicts). Entries A1–A10 entries are not isolated; they

[15]https://olafwitkowski.com/2025/03/11/ai-to-ai-communication-unpacking-gibberlink-secrecy-and-new-ai-communication-channels/

Table 8. Challenges and Opportunities for GenAI Copilots

Form of GenAI augmentation	Challenge	Opportunity
A1. Prompt engineering and specification clarity	How to systematically design prompts or APIs for GenAI elements so outputs are reliable, traceable, and modifiable?	Create formal abstractions or domain-specific prompt languages that can be composed, validated, and versioned in software pipelines.
A2. Traceability, provenance, rationale, and audit trails	When a GenAI tool produces, e.g., code snippets, designs, test cases, how do you trace which input tokens or model internals led to it?	Embedding provenance metadata or "explanation traces" in software artifacts to enable accountability, debugging, and compliance.
A3. Quality control & validation of AI-generated artifacts	Ensuring that AI-generated process artifacts (e.g., requirements drafts, design sketches, code suggestions) are correct, consistent, non-contradictory, and align with system constraints.	Hybrid verification approaches (symbolic + statistical) to validate, constrain, or filter AI outputs before acceptance.
A4. Model drift, update, version management, and maintenance of GenAI elements	Over time, the context, codebase, or domain evolves; the GenAI elements must be re-aligned. How to manage this versioning, continuous learning, and validation?	Develop continuous integration pipelines for AI models (CI/AI), similar to CI/CD, to monitor performance, retrain, revoke, or rollback when degradation occurs.
A5. Workflow integration and orchestration	Embedding GenAI process tools into conventional development workflows (CI, issue trackers, code review) without disrupting developer flow.	Research orchestration layers or microservices that interleave GenAI calls transparently in Dev pipelines, handling context, caching, and incremental updates.
A6. Cognitive load, transparency, and user control	Even as GenAI offers suggestions, developers must remain in control. Too much opacity or "black-box" behavior may erode trust or lead to overreliance.	Design user interfaces and interaction modalities (e.g., "explain why this suggestion") that balance automation and human oversight effectively.
A7. Bias, fairness, and domain alignment	The training data or model biases may propagate into process artifacts (requirements, testcases, design decisions).	Techniques to detect, mitigate, or "debias" generated outputs in domain-specific engineering settings (e.g., safety-critical, regulated systems).
A8. Collaborative work and multi-user context merging	When multiple developers or stakeholders interact with GenAI suggestions in parallel, merging outputs can lead to conflicts. Individual contexts (e.g., changes, actions, perspectives) need to be properly merged or synchronized.	Techniques to merge multiple suggestion threads, version control for AI outputs, context-aware collaborative "AI-augmented" tools capable of merging changes so that the final result makes sense to everyone, ensuring everyone sees a consistent view of the shared work.
A9. Safety, security, & dependency risk	If the GenAI element itself becomes a dependency (e.g., an external API), what happens if it fails, changes, or is compromised?	Formalizing fallback strategies, sandboxing, and verification for AI-assisted process tools.
A10. Trustworthy GenAI	How to build GenAI elements so that outputs are trustworthy?	Identify even less powerful, generic, and constrained GenAI that is trustworthy under specific assumptions and in specific contexts.

form an interconnected set of challenges that require both engineering and organizational solutions. Only through controlled experimentation, empirical evidence, and collaboration across software engineering, machine learning, HCI, ethics, and regulatory disciplines will it be possible to turn these opportunities into truly useful, reliable, and adoptable GenAI Copilots for software development. This section therefore closes with a call to pursue the operational directions outlined above, balancing technical innovation with socio-technical governance to build copilots that assist, rather than replace, human judgment and responsibility.

Table 9. Mapping between forms of GenAI augmentation and its sources for GenAI Copilots

Research Challenges and Opportunities	Source (McLuhan's tetrad)
A1. Prompt engineering and specification clarity	Trustworthiness and Reliability of the Reverses quadrant.
A2. Traceability, provenance, rationale, and audit trails	Explainability of the Reverses quadrant.
A3. Quality control & validation of AI-generated artifacts	Compliance of the Reverses quadrant.
A4. Model drift, update, and maintenance of GenAI elements	Compliance of the Reverses quadrant.
A5. Workflow integration and orchestration	Strict boundaries between SDLC stages of the Reverses quadrant.
A6. Cognitive load, transparency, and user control	Accountability and Trustworthiness & Reliability of the Reverses quadrant
A7. Bias, fairness, and domain alignment	Fairness of the Reverses quadrant.
A8. Collaborative editing and multi-user context merging	Project & Process Management and Human & Team Factors of the Enhances quadrant.
A9. Safety, security, & dependency risk	Security of the Reverses quadrant.
A10. Trustworthy GenAI	Trustworthiness and Reliability of the Reverses quadrant. Runtime Verification and Validation of the Retrieves quadrant.

8.2 Roadmap Part B: GenAI Teammates

Table 10 outlines the central challenges and opportunities in augmenting the process via autonomous agents (GenAI Teammates). The issues primarily concern the balance between autonomy and human oversight (B1), conflict resolution in multi-agent or human–agent teams (B2), and the ability to engage in long-horizon, multi-step planning across development phases (B3). Questions of accountability and responsibility attribution (B4) as well as mechanisms for recovery, rollback, and self-diagnosis (B5) highlight the need for formal governance and resilience strategies. Additional challenges include coordination across heterogeneous agents operating on different parts of the software lifecycle (B6) and the calibration of human trust in agent recommendations (B7). The opportunities point to the development of supervisory architectures, negotiation protocols, embedded planning capabilities, formal responsibility frameworks, self-monitoring mechanisms, and coordination protocols, all aimed at creating safe, transparent, and effective human-agent collaboration in complex engineering workflows.

Table 11 maps the research challenges and opportunities to the source that triggered the form of GenAI Teammate augmentation. As above, the source can be one or more elements in the corresponding tetrad. The mapping to McLuhan's tetrad highlights that ethical and autonomy concerns predominate and should shape design choices. Research should therefore focus on supervisory architectures, meta-control policies, negotiation and consensus protocols, embedded planning modules, and contract-like responsibility models. Complementary engineering work must deliver runtime verification, safe fallback strategies, and shared knowledge protocols for inter-agent consistency. Finally, multidisciplinary evaluation, combining controlled experiments, field studies, and governance frameworks, is essential to produce human-agent teams that are effective, accountable, and socially acceptable.

8.3 Roadmap Part C: GenAIware

Table 12 describes the GenAIware aspects by highlighting deployment-centric challenges and opportunities for integrating GenAI into software development and operation.

Table 10. Challenges and Opportunities for GenAI Teammates

Form of GenAI augmentation	Challenge	Opportunity
B1. Agent autonomy vs human oversight balance	Deciding when the agent should act autonomously versus requesting human confirmation; how to avoid undesirable emergent behavior.	Develop meta-control policies or supervisory architectures that mediate trust, escalation, and safe boundaries for agent actions.
B2. Conflict resolution and negotiation among agents & humans	In a team setting, multiple agents (or agent + human) may propose conflicting plans or changes. How to mediate conflicts?	Research negotiation protocols, consensus algorithms, or hierarchical control structures for hybrid teams.
B3. Long-horizon planning, multi-step decision-making	Agents must plan across multiple SDLC phases, not just local suggestions. This requires prediction, cost estimation, tradeoff reasoning.	Agents with embedded planning modules, reinforcement learning for planning over development workflows, and cost-benefit reasoning.
B4. Accountability and responsibility attribution	When an agent autonomously makes a change that fails or introduces defects, who is accountable?	Formal frameworks for attributing responsibility in human-AI collaborative systems, perhaps with contract models or "permission scoping".
B5. Recovery, rollback, and agent self-diagnosis	Agents must detect when their own suggestions degrade system quality or violate constraints, and autonomously rollback or correct.	Research self-monitoring agents that propose and validate undo plans, safe failsafes, and "what-if" scenario testing.
B6. Inter-agent coordination across modules	If multiple agents operate (e.g., one for requirements, one for testing, one for deployment), they must coordinate and exchange context.	Architectures for agent collaboration, shared knowledge bases, or protocols for consistency and dependency handling.
B7. Trust calibration and human-agent teaming	Humans must calibrate when to rely on agent proposals. Too much trust leads to overreliance, too little undermines utility.	Techniques to signal confidence, uncertainty, or explain rationale to support human trust calibration.

Table 11. Mapping between forms of GenAI augmentation and its sources for GenAI Teammates

Research Challenges and Opportunities	Source (McLuhan's tetrad)
B1. Agent autonomy vs human oversight balance	Ethics and autonomy of the Reverses quadrant.
B2. Conflict resolution and negotiation among agents & humans	Ethics and autonomy of the Reverses quadrant.
B3. Long-horizon planning, multi-step decision-making	Ethics and autonomy of the Reverses quadrant.
B4. Accountability and responsibility attribution	Accountability & Code Ownership of the Reverses quadrant.
B5. Recovery, rollback, and agent self-diagnosis	Code Quality and Technical Debt of the Retrieves quadrant. Static Analysis of the Retrieves quadrant.
B6. Inter-agent coordination across modules	Ethics and autonomy of the Reverses quadrant.
B7. Trust calibration and human-agent teaming	Ethics and autonomy of the Reverses quadrant.

The table emphasizes the need for adaptive prompt mechanisms that adjust dynamically to runtime contexts (C1), along with rigorous approaches to testing and validation to assure the correctness and safety of GenAI-infused features (C2). Performance considerations such as inference cost, latency, and resource efficiency (C3) must be balanced with user expectations for reliability and robust fallback behavior when outputs fail (C4). Security and privacy concerns are critical, as embedded models may inadvertently expose sensitive data (C5). Sustaining these systems further requires lifecycle management strategies, including safe in-field updates, controlled A/B testing, and dynamic rollback capabilities (C6). Finally, explainability and transparency (C7) are essential for both

Table 12. Challenges and Opportunities for GenAIware

Form of GenAI augmentation	Challenge	Opportunity
C1. Dynamic prompt engineering in deployed systems	At runtime, code must generate appropriate prompts or adaptively modify them as user input context changes.	Self-tuning prompt modules or meta-prompting layers that adapt to usage distributions and context drift.
C2. Testing, verification, and validation of GenAI-infused features	Ensuring that the GenAI-enabled portions of software meet functional, nonfunctional, and safety requirements.	New testing paradigms, formal specifications for GenAI behaviors.
C3. Performance, latency, and resource constraints in deployment	Embedding GenAI into products often raises concerns of inference cost, latency, memory, and energy.	Edge-compatible GenAI modules, hybrid local/-cloud partitioning, caching, and approximate inference.
C4. User expectation, reliability, and backoff strategies	Users often expect predictable behavior; when GenAI fails or returns suboptimal output, the product must cope gracefully.	Fallback rules, ensemble strategies, confidence thresholds, graceful degradation paths, human-in-the-loop fallback.
C5. Security, privacy, and data leakage in GenAI elements	The embedded GenAI might memorize or leak private or sensitive data.	Techniques like differential privacy, context-window isolation, on-device fine-tuning, or encryption to mitigate leakage.
C6. ML Model lifecycle in-field updates and A/B testing	Updating or deploying new machine learning models inside a product, measuring the effect on users online, dynamically rolling back if negative side effects are detected.	Tools for safe online A/B testing of GenAI submodules, rollback strategies, incremental updates, and versioning at runtime to actively maintain and improve a model after deployment, using real-world data and real-time feedback to keep it performing well over time.
C7. Explainability and transparency within features	Users and regulators may require explanations.	Embed explanation layers or rationale surfaces in the product, enabling "why this response" queries or tracebacks.

user trust and regulatory compliance. Collectively, the opportunities point toward adaptive prompting layers, new verification paradigms, efficient deployment strategies, privacy-preserving techniques, robust lifecycle tooling, and explanation features that enable reliable and trustworthy GenAIware in the software development lifecycle.

Table 13. Mapping between forms of GenAI augmentation and its sources for GenAIware

Research Challenges and Opportunities	Source (McLuhan's tetrad)
C1. Dynamic prompt engineering in deployed systems	Prompt Programming of the Enhances quadrant.
C2. Testing, verification, and validation of GenAI-infused features	Responsible and Secure AI of the Enhances quadrant. Runtime Verification and Validation of the Retrieves quadrant.
C3. Performance, latency, and resource constraints in deployment	Efficiency and Sustainability of the Reverses quadrant.
C4. User expectation, reliability, and backoff strategies	Reliability of the Reverses quadrant.
C5. Security, privacy, and data leakage in GenAI elements	Responsible and Secure AI of the Enhances quadrant.
C6. ML Model lifecycle in-field updates and A/B testing	Traditional Static Testing, and Separation between Machine Learning Cycle and SDLC of the Obsolesces quadrant.
C7. Explainability and transparency within features	Opacity of AI elements of the Obsolesces quadrant.

As already done for the GenAI Copilot and GenAI Teammate augmentations, Table 13 maps the research challenges and opportunities to one or more tetrad elements in Section 6 that triggered GenAI GenAIware in SDLC. In short, Table 12 and Table 13 frame GenAIware as an integration-, embedding- and deployment-centered challenge: adaptive prompting, rigorous validation, performance and fallback engineering, security/privacy, lifecycle updates, and explainability are all essential. Key opportunities include adaptive prompting layers, new verification paradigms, efficient deployment strategies, privacy-preserving techniques, robust lifecycle tooling, and rich explanation features. Also, for this type of augmentation, design should be informed by the tetrad mapping so that enhancements do not inadvertently reverse or obscure critical system properties. Top research priorities are runtime-aware prompt controllers that adapt to context and user intent, and scalable testing/validation frameworks for generative features. Engineering work must address cost/latency trade-offs (model selection, quantization, caching) while guaranteeing reliable fallback behavior. Security and privacy require privacy-by-design, data minimization, and provenance controls to avoid sensitive exposure. Operational tooling-observability, A/B pipelines, and automated rollback will be crucial for safe in-field evolution. Finally, multidisciplinary evaluation (benchmarks, field studies, regulatory alignment) is needed to ensure GenAIware is performant, trustworthy, and compliant in real-world software products.

8.4 Roadmap Part D: GenAI Robots

Table 14 describes challenges and opportunities of GenAI Robots in SE. The table focuses on the challenges and opportunities of engineering software systems where part of the functionality is delivered by autonomous GenAI agents. Key risks include emergent behavior and goal drift over time (D1), the need to preserve user trust and control through override or reversibility mechanisms (D2), and the stability of agents that adapt or learn in the field without violating constraints (D3). Additional concerns arise when multiple agents interact in shared environments, where competition or interference may disrupt outcomes (D4), and when scaling such systems to production settings introduces significant infrastructure and cost challenges (D5). The opportunities point toward research in alignment and auditing techniques, user-facing control interfaces, safe online learning methods, multi-agent coordination protocols, and efficient architectures that minimize computational overhead while maintaining robust performance.

Table 15 completes the mapping of the research challenges and opportunities referring to the tetrad of the fourth form of augmentation in Section 7. In summary, the integration of GenAI Robots into the SDLC brings tangible challenges and related risks, such as emergent behaviors and goal drift, loss of control, instability in field learning, multi-agent interference, and scalability costs, but also clear opportunities for software engineering research. This section highlights that, also in this case, solutions are not isolated: they require alignment and auditing methods, human control interfaces and reversibility, safe online learning techniques, multi-agent coordination protocols, and compute-efficient architectures. Progress demands a roadmap that combines theoretical research, field experiments, and engineering practices (testing, metrics, and governance) to measure and mitigate risk. Only a systemic and interdisciplinary approach can turn these autonomous agents from risk factors into reliable enablers of the software development process.

8.5 Roadmap Part X: Cross-form Aspects

Tables 16 and 17 describe cross-form challenges and opportunities (X1–X8) that extend beyond individual workflows or deployment contexts. X1 highlights the complexity of coordinating process-level and product-level agents, while X2 stresses the absence of robust benchmarks and evaluation methods for measuring GenAI's impact in engineering practice. Human factors emerge prominently in X3, where changes to team roles, skills, and dynamics require careful study, and in X4, where unresolved questions of intellectual property, licensing, and liability remain. Broader systemic concerns include sustainability and environmental costs (X5), resilience against

Table 14. Challenges and Opportunities for GenAI Robots

Form of GenAI augmentation	Challenge	Opportunity
D1. Emergent behavior, alignment, and goal drift	Over time, autonomous product agents may stray from their intended goals or produce unexpected behavior under edge conditions.	Research alignment techniques, safe sandbox testing, reward shaping, and continual auditing of behavior drift.
D2. User trust, control, and override mechanisms	As the product acts more autonomously, users must retain ultimate control; the agent must allow override, reversibility, and "undo."	User interfaces for control granularity, adjustable autonomy levels, interactive consent, and transparency.
D3. Evolution, self-improvement, and adaptation	Agents that learn in the field must adapt without compromising stability or violating constraints.	Techniques for safe online learning, constraint enforcement, minimizing "catastrophic forgetting" and ensuring consistency.
D4. Inter-agent competition or collaboration in a shared environment	If multiple intelligent agents act and interact (e.g., multiple AI users on same platform), coordination, conflicts, or negative interference may arise.	Multi-agent coordination algorithms, negotiation protocols, hierarchical control, or regulatory "protocols" for agent behavior.
D5. Scalability, infrastructure, and operational cost	Running autonomous agents (with reasoning, evaluation, planning) at scale in production environments is expensive.	Efficient architectures, caching, hierarchical reasoning, selective activation, and event-triggered execution to reduce compute load.

Table 15. Mapping between forms of GenAI augmentation and its sources for GenAI Robots

Research Challenges and Opportunities	Source (McLuhan's tetrad)
D1. Emergent behavior, alignment, and goal drift	Novel forms of human-computer interaction (HCI) of the Enhances quadrant. Human-agent balance and Traditional separation of concerns of the Reverses quadrant.
D2. User trust, control, and override mechanisms	Human-agent balance and Traditional separation of concerns of the Reverses quadrant.
D3. Evolution, self-improvement, and adaptation	Novel forms of human-computer interaction (HCI) of the Enhances quadrant. Static assurance of legal and ethical norms of the Reverses quadrant.
D4. Inter-agent competition or collaboration in a shared environment	Novel forms of human-computer interaction (HCI) of the Enhances quadrant. Human-agent balance of the Reverses quadrant.
D5. Scalability, infrastructure, and operational cost	Traditional separation of concerns of the Reverses quadrant.

adversarial manipulation (X6), and the limited transferability of GenAI methods across domains (X7). Finally, X8 underscores the need for continuous adaptation as software ecosystems evolve. Together, these themes point to a research agenda that blends technical innovation with organizational, legal, and ethical considerations to enable trustworthy GenAI augmentation.

9 Threats to Validity

This section discusses potential threats to the validity of our study and the measures taken to mitigate them.

Construct Validity. Construct validity concerns whether the key concepts investigated were correctly defined and operationalized. A potential threat lies in the subjective interpretation of what constitutes GenAI augmentation in software engineering processes and artifacts. To address this, the fundamental forms of GenAI in SE were initially elicited through collective discussions during the FSE 2025 "2030 Software Engineering" workshop

Table 16. Challenges and Opportunities for Cross-form GenAI Augmentation

Form of GenAI augmentation	Challenge	Opportunity
X1. Hybrid systems: integrating process agents + product agents	In future systems, process-level agents (that plan development) and product-level agents (in software) may co-exist and influence each other. Managing their interactions is complex.	Architectures and protocols for cross-layer consistency, feedback loops, negotiation, and coordinated evolution.
X2. Metrics, evaluation benchmarks, and empirical validation	How do we measure "goodness" of GenAI augmentation (for productivity, reliability, maintainability, human satisfaction)?	Define standard benchmarks, controlled experiments, and evaluation suites for hybrid human-AI engineering settings.
X3. Human factors, team dynamics, education and roles	As GenAI changes workflows, roles and skills shift. Resistance, cognitive burden, misuse, or misalignment may arise.	Study human-AI teaming, change management, training curricula, and social/organizational adaptation strategies.
X4. Intellectual property, licensing, and legal liability	Who owns code, designs, or artifacts generated by GenAI? What licenses apply? What liability when AI-generated code fails?	Legal and policy frameworks for IP attribution, shared licensing models, AI-generated artifact contracts, and compliance regimes.
X5. Model robustness, adversarial attacks, and misuse	GenAI elements and agents can be manipulated via adversarial prompts or injection attacks to produce harmful outputs.	Robustness research, prompt sanitization, query filtering, anomaly detection, and secure input/output boundaries.
X6. Updating of knowledge, evolving software landscapes	The software ecosystem (languages, frameworks, libraries) evolves rapidly; GenAI augmentation must keep up.	Continual learning, modular plugin adaptation, community-based model updates, and open model ecosystems in the software engineering domain.

and grounded in existing taxonomies, such as the one proposed by SEI. These constructs were subsequently refined and validated through multiple iterations of internal discussions and external feedback sessions involving peers. This triangulation of sources and perspectives aimed to ensure that the constructs reflect a shared and well-grounded understanding rather than the isolated view of a subset of authors.

Another potential threat concerns the coverage of the Rapid Literature Reviews (RLRs) used in Design Cycle 2. The search strings adopted in each review may not have captured all relevant literature, potentially limiting the breadth of the investigated evidence. Moreover, the use of LLM-based filtering (using Qwen3 and Gemini-2.5 respectively) might have inadvertently excluded relevant publications due to the potential risks of bias and hallucination. To mitigate this validity risk, we manually inspected samples of the excluded papers to double check that no relevant literature was systematically missed. We also provide open access to the data of our literature reviews [17], thereby providing additional transparency and replicability of the RLR process.

Internal Validity. Internal validity refers to the soundness of the study design and the correctness of the reasoning linking evidence to conclusions. A potential risk arises from interpretive bias during the construction of the McLuhan Tetrads and from the compressed time frame of the rapid reviews, which could have reduced the depth of evidence interpretation. To mitigate these risks, the research was organized into three design science cycles, each including explicit awareness, solution, and validation stages. Within Cycle 2, the RLRs followed a shared baseline protocol and were independently executed by different author teams. Subsequent cross-review by authors not involved in their initial construction helped reduce confirmation bias and ensured consistency and traceability across the four forms of GenAI.

External Validity. External validity concerns the generalizability and transferability of the study results. Since the goal of this work was to design a conceptual roadmap rather than to produce statistically generalizable

Table 17. Mapping between Cross-form Research Challenges and Opportunities

Cross-cutting Research Challenges and Opportunities	**Source (Process)**	**Source (Product)**
X1. Hybrid systems: integrating process agents + product agents	A5. Workflow integration and orchestration. B6. Inter-agent coordination across modules.	D4. Inter-agent competition or collaboration in a shared environment.
X2. Metrics, evaluation benchmarks, and empirical validation	A10. Trustworthy GenAI	C4. User expectation, reliability, and backoff strategies. C6. ML Model lifecycle in-field updates and A/B testing.
X3. Human factors, team dynamics, education and roles	A6. Cognitive load, transparency, and user control. B2. Conflict resolution and negotiation among agents & humans. B7. Trust calibration and human-agent teaming.	C7. Explainability and transparency within features. D2. User trust, control, and override mechanisms.
X4. Intellectual property, licensing, and legal liability	A6. Cognitive load, transparency, and user control. B4. Accountability and responsibility attribution.	C7. Explainability and transparency within features. D2. User trust, control, and override mechanisms.
X5. Model robustness, adversarial attacks, and misuse	A9. Safety, security, & dependency risk. B5. Recovery, rollback, and agent self-diagnosis.	C2. Testing, verification, and validation of GenAI-infused features. C5. Security, privacy, and data leakage in GenAI elements.
X6. Updating of knowledge, evolving software landscapes	A4. Model drift, update, and maintenance of GenAI elements.	D3. Evolution, self-improvement, and adaptation

evidence, its external validity is inherently limited. However, its findings draw upon diverse forms of evidence, including literature, workshop discussions, and multi-author evaluations, which provide analytical rather than statistical generalization. Furthermore, the final roadmap was refined through iterative co-author reviews and three dedicated meetings, strengthening confidence in the broader relevance and applicability of the identified challenges and opportunities.

Conclusion Validity. Conclusion validity relates to the credibility, logical coherence, and consistency of the study's outcomes. Potential threats include overgeneralization and interpretive divergence among authors. To address these, all results were progressively consolidated and validated across the three design science cycles. In the final Cycle 3, two authors synthesized the validated tetrads into the roadmap, while the remaining authors collectively reviewed and refined it until consensus was reached. This iterative synthesis and collective validation ensured that the conclusions drawn in the roadmap are well supported by the evidence gathered and consistently interpreted within the scope of the study.

10 Conclusion and Perspectives

GenAI implies a paradigm shift in SE, profoundly reshaping both development processes and the nature of software products. To navigate this complex transformation, we introduced a systematic framework for understanding and analyzing the impact of GenAI on SE. We proposed a structuring of GenAI augmentation along two principal dimensions: what is being augmented (process versus product) and the level of autonomy of the augmentation (passive versus active). This categorization yields four distinct and tangible forms of GenAI augmentation: the GenAI Copilot, GenAIware, the GenAI Teammate, and the GenAI Robot.

Using this framework as our analytical lens, we conducted an examination of the state-of-the-art for each of the four forms. Our methodology – grounded in a multi-cycle design science approach – employed rapid literature reviews to gather evidence, which was then synthesized using McLuhan's tetrads. This approach enabled a holistic assessment of each form of GenAI augmentation, identifying what it enhances, what established practices it reverses when pushed to extremes, what past concepts it retrieves, and what it may render obsolete. This multidimensional analysis revealed a complex interplay of benefits and challenges, from enhanced productivity and retrieved formalisms and paradigms to issues such as trustworthiness, accountability, and the potential erosion of human expertise.

The culmination of this analysis is a comprehensive research roadmap that distills our findings into specific, actionable challenges and opportunities. This roadmap addresses not only the unique issues pertinent to each of the four forms but also identifies cross-cutting themes, including human-AI team dynamics, legal and ethical considerations, and the need for robust evaluation benchmarks.

By offering a structured and evidence-based perspective, this work refines and updates existing roadmaps, serving as part of an overall prescriptive guide for the SE community contained within this special issue. As GenAI continues its rapid evolution, this roadmap serves as a foundation for future research, encouraging a balanced approach that maximizes the transformative potential of GenAI while proactively mitigating its inherent risks, thereby helping to shape the future principles, practices, and economics of software engineering.

—

We like to close by offering our **ten predictions for software engineering in the year 2030**:

Death of manual coding for routine tasks: By 2030, 70% of boilerplate code, CRUD operations, and standard API integrations will be autonomously generated by GenAI Teammates with zero human keystrokes. Human developers will spend less than 20% of their time writing code from scratch. Our roadmap indicates that manual coding will become obsolete. The shift from developer-as-coder to developer-as-architect/orchestrator will be complete for routine functionality.

Prompt Engineering as a core SE discipline: By 2030, "Prompt Architect" will be a recognized job title with professional certifications. Universities will offer dedicated courses on prompt engineering, and 40% of SE job postings will list prompt engineering as a required skill alongside traditional programming languages. Our roadmap indicates that prompt programming will be a new "programming" paradigm. As GenAIware proliferates, prompt quality directly determines system reliability.

Emergence of "Software Compliance as Code": By 2030, legal compliance, ethical constraints, and safety requirements will be encoded as formal guardrails in machine-readable formats (e.g., "compliance specification languages") that GenAI systems must satisfy in real-time. 40% of regulated software projects will use automated compliance verification that checks both human and AI contributions against these formal specifications. The EU AI Act and similar regulations will mandate this for high-risk AI systems. Our roadmap mentions static assurance of legal/ethical norms becoming insufficient together with IP, licensing, and legal liability, which will lead to widespread concerns about compliance, fairness, and responsible AI across all forms. The need for guardrails that work at runtime, not just design time, is critical.

Rise of AI accountability standards: By 2030, at least three major organizations (ACM, IEEE, or ISO) will have published formal standards for AI accountability in software development, contributing to the aforementioned "software compliance as code". 50% of Fortune 500 companies will require explicit "AI Attribution Metadata" in their codebases, documenting which code was human-written, AI-suggested, or AI-autonomous. This results from our roadmap indicating widespread concerns about accountability and code ownership.

Emergence of "Responsible GenAI Engineering": By 2030, there will be dedicated "GenAI Responsibility Engineers" on software teams, who are in charge of implementing guardrails, detecting hallucinations, and ensuring ethical and sustainable AI behavior. Responsible engineering for GenAI will become distinct from

traditional QA, with specialized tools and methodologies. At least 20% of SE teams in regulated industries will have this role. This is backed by our roadmap that indicates the relevance of testing/verification of GenAI features, security and privacy, emergent behavior and goal drift detection, and adversarial attacks, which raises concerns about reliability and trustworthiness.

Obsolescence of Classical IDEs: By 2030, traditional IDEs (such as VS Code, IntelliJ) will face existential competition from "conversational development environments" where 30% of developers primarily interact with their codebase through natural language chat interfaces rather than file editors. The concept of "manually opening and editing files" becomes optional for many tasks. This becomes evident from our roadmap as GenAI Teammates will mean that traditional IDE editors fade, emphasizing natural language interaction as the primary interface between humans and AI.

Integrated CI/CD/AI pipelines: By 2030, CI/CD will evolve into "CI/CD/AI" where 80% of software projects include automated pipelines for monitoring, retraining, and rolling back GenAI components. Model drift detection and automated A/B testing of GenAI features will be as common as unit testing is today. Our roadmap clearly indicates the consideration of model drift and maintenance, AI model lifecycle updates, and the need for runtime verification and validation.

Multi-Agent orchestration platforms: By 2030, at least 5 major platforms for orchestrating multiple GenAI agents (both Teammates and Robots) will have emerged, with at least one achieving "Kubernetes-level" adoption (used by 40%+ of organizations). These platforms will standardize agent-to-agent communication protocols, making multi-agent systems as manageable as microservices are today. This is backed by our roadmap indicating emerging inter-agent coordination, multi-agent competition/collaboration, as well as hybrid systems integrating process and product agents.

"10x Developer" becomes "10x Orchestrator": By 2030, the concept of the "10x developer" will shift from someone who writes 10x more code to someone who effectively orchestrates 10+ GenAI agents simultaneously. Performance metrics will measure "agents successfully coordinated" rather than "lines of code written." Top performers will manage complex ecosystems of specialized AI agents rather than large codebases. Our roadmap reflects this transformation across all forms, particularly with respect to agent autonomy balance, trust calibration, and the general shift from coding to orchestration.

GenAI-induced technical debt crisis and recovery: By 2030, 60% of software maintenance effort will focus on understanding and improving AI-generated code rather than human-written code. In particular, we will see a new generation of "AI Code Archaeology" tools that can analyze, explain, and refactor legacy AI-generated code. This is to address the development of previous years where industry will experience a "GenAI technical debt crisis", during which accumulated low-quality AI-generated code creates major maintenance problems. Our roadmap indicates this by mentioning quality control of AI artifacts and maintenance of GenAI elements, together with code comprehension, code quality and static analysis.

Acknowledgments

Some parts of our work were assisted by Generative AI. We used Elsevier's ScopusAI, Google's Gemini-2.5, and Alibaba's Qwen3 during the rapid literature review (also see Appendix B). We also used Anthopic's Claude Sonnet 4.5 to help us brainstorm the concluding predictions for software engineering in the year 2030.

We thank Abhik Roychoudhury and Mauro Pezze – co-chairs of the FSE 2025 workshop on "Software Engineering 2030" – for their brilliant organization of such an encouraging and stimulating workshop, as well as to all fellow workshop participants for the engaging discussions.

This work was partly funded by Core Informatics at KIT (KiKIT) and Topic Engineering Secure Systems of the Helmholtz Association (HGF) and partly supported by the German Research Foundation (DFG) - SFB 1608 - 501798263 and KASTEL Security Research Labs, as well as (a) the MUR (Italy) Department of Excellence 2023–2027,

(b) the European HORIZON-KDT-JU research project MATISSE “Model-based engineering of Digital Twins for early verification and validation of Industrial Systems” (101140216), (c) the PRIN project P2022RSW5W - RoboChor: Robot Choreography, (d) the PRIN project 2022JKA4SL - HALO: etHical-aware AdjustabLe autOnomous systems, and (e) the PRIN project 2022JAFATE - CAVIA: enabling the Cloud-to-Autonomous-Vehicles continuum for future Industrial Applications; (f) the European Union - NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant ECS00000041 - VITALITY – CUP: D13C21000430001; (g) PNRR MUR (Italy) – Centro Nazionale HPC, Big Data e Quantum Computing, Spoke9 - Digital Society & Smart Cities (grant CN_00000013).

References

[1] Silvia Abrahão, John Grundy, Mauro Pezzè, Margaret-Anne Storey, and Damian A. Tamburri. 2025. Software Engineering by and for Humans in an AI Era. ACM Trans. Softw. Eng. Methodol. 34, 5, Article 129 (May 2025), 46 pages. doi:10.1145/3715111

[2] Iftekhar Ahmed, Aldeida Aleti, Haipeng Cai, Alexander Chatzigeorgiou, Pinjia He, Xing Hu, Mauro Pezzè, Denys Poshyvanyk, and Xin Xia. 2025. Artificial Intelligence for Software Engineering: The Journey So Far and the Road Ahead. ACM Trans. Softw. Eng. Methodol. 34, 5 (2025), 119:1–119:27. doi:10.1145/3719006

[3] Mamdouh Alenezi and Mohammed Akour. 2025. AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. Applied Sciences 15 (2025). Issue 3. https://doi.org/10.3390/app15031344

[4] Zahra Aminiranjbar, Jianan Tang, Qiudan Wang, Shubha Pant, and Mahesh Viswanathan. 2025. DAWN: Designing Distributed Agents in a Worldwide Network. IEEE Access 13 (2025), 138795–138812.

[5] Timothy J. Aveni, James Smith, Armando Fox, and Björn Hartmann. 2025. Supporting Students in Prototyping AI-backed Software with Hosted Prompt Template APIs. In ITiCSE (1). ACM, 65–71.

[6] Aneesha Bakharia. 2025. Iterative Proof-Driven Development LLM Prompt. In WWW (Companion Volume). ACM, 1596–1597.

[7] Leonardo Banh, Florian Holldack, and Gero Strobel. 2025. Copiloting the Future: How Generative AI Transforms Software Engineering. Inf. Softw. Technol. 183, C (June 2025). doi:10.1016/j.infsof.2025.107751

[8] Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven Failure Points When Engineering a Retrieval Augmented Generation System. In CAIN. ACM, 194–199.

[9] Richard Baskerville, Abayomi Baiyere, et al. 2018. Design science research contributions: Finding a balance between artifact and theory. Journal of the Association for Information Systems 19, 5 (2018).

[10] Marco Becattini, Roberto Verdecchia, and Enrico Vicario. 2025. SALLMA: A Software Architecture for LLM-Based Multi-Agent Systems. In SATrends@ICSE. IEEE, 5–8.

[11] Abhiram Bellur and Fraol Batole. 2025. IDE Native, Foundation Model Based Agents for Software Refactoring. In 2025 IEEE/ACM Second IDE Workshop (IDE). 42–45. doi:10.1109/IDE66625.2025.00013

[12] Islem Bouzenia, Premkumar Devanbu, and Michael Pradel. 2024. Repairagent: An autonomous, llm-based agent for program repair. arXiv preprint arXiv:2403.17134 (2024).

[13] Alessio Bucaioni, Martin Weyssow, Junda He, Yunbo Lyu, and David Lo. 2025. A Functional Software Reference Architecture for LLM-Integrated Systems. In ICSA Companion. IEEE, 1–5.

[14] Bruno Cartaxo, Gustavo Pinto, and Sergio Soares. 2018. The Role of Rapid Reviews in Supporting Decision-Making in Software Engineering Practice. In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 (Christchurch, New Zealand) (EASE ’18). Association for Computing Machinery, New York, NY, USA, 24–34. doi:10.1145/3210459.3210462

[15] Jianhui Chen and Yunchao Peng. 2025. Development of AI Agent Based on Large Language Model Platforms. In 2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD). 864–868. doi:10.1109/ICAIBD64986.2025.11082094

[16] Lukasz Chomatek, Wojciech Slabosz, and Aneta Poniszewska-Maranda. 2025. From Words to Wisdom: LLMs Summarizing Instructional Content. In Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering, FSE Companion 2025, Clarion Hotel Trondheim, Trondheim, Norway, June 23-28, 2025, Leonardo Montecchi, Jingyue Li, Denys Poshyvanyk, and Dongmei Zhang (Eds.). ACM, 1623–1630. doi:10.1145/3696630.3728697

[17] Andreas Metzger (contact person). 2026. A Research Roadmap for Augmenting Software Engineering Processes and Software Products with Generative AI: Rapid Literature Review Results. doi:10.5281/zenodo.18345896

[18] Fiona Draxler, Anna Werner, Florian Lehmann, Matthias Hoppe, Albrecht Schmidt, Daniel Buschek, and Robin Welsch. 2024. The AI ghostwriter effect: When users do not perceive ownership of AI-generated text but self-declare as authors. ACM Transactions on Computer-Human Interaction 31, 2 (2024), 1–40.

[19] Zhuoyun Du, Chen Qian, Wei Liu, Zihao Xie, Yifei Wang, Rennai Qiu, Yufan Dang, Weize Chen, Cheng Yang, Ye Tian, et al. 2025. Multi-Agent Collaboration via Cross-Team Orchestration. In Findings of the Association for Computational Linguistics: ACL 2025.

10386–10406.

[20] Evelyn Duesterwald, Vatche Isahagian, K. R. Jayaram, Ritesh Kumar, Vinod Muthusamy, Punleuk Oum, Gegi Thomas, and Praveen Venkateswaran. 2024. A Conversational Assistant Framework for Automation. In Proceedings of the 25th International Middleware Conference Industrial Track, Middleware Industrial Track 2024, Hong Kong, SAR, China, December 2-6, 2024. ACM, 1–7. doi:10.1145/3700824.3701093

[21] Timothy Elvira, Tyler Thomas Procko, and Omar Ochoa. 2024. Requirements Elicitation for Machine Learning Applications: A Research Preview. In 2024 Conference on AI, Science, Engineering, and Technology (AIxSET). 218–221. doi:10.1109/AIxSET62544.2024.00042

[22] Daniel Erhabor, Sreeharsha Udayashankar, Meiyappan Nagappan, and Samer Al-Kiswany. 2025. Measuring the Runtime Performance of C++ Code Written by Humans Using Github Copilot. In 47th IEEE/ACM International Conference on Software Engineering, ICSE 2025, Ottawa, ON, Canada, April 26 - May 6, 2025. IEEE, 2062–2074. doi:10.1109/ICSE55347.2025.00059

[23] Matteo Esposito, Xiaozhou Li, Sergio Moreschini, Noman Ahmad, Tomás Cerný, Karthik Vaidhyanathan, Valentina Lenarduzzi, and Davide Taibi. 2025. Generative AI for Software Architecture. Applications, Trends, Challenges, and Future Directions. SSRN (2025). https://dx.doi.org/10.2139/ssrn.5196419

[24] Wenqi Fan, Pangjing Wu, Yujuan Ding, Liangbo Ning, Shijie Wang, and Qing Li. 2025. Towards Retrieval-Augmented Large Language Models: Data Management and System Design . In 2025 IEEE 41st International Conference on Data Engineering (ICDE). IEEE Computer Society, Los Alamitos, CA, USA, 4509–4512. doi:10.1109/ICDE65448.2025.00341

[25] Li Feng, Ryan Yen, Yuzhe You, Mingming Fan, Jian Zhao, and Zhicong Lu. 2024. CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural Language Programming. In CHI. ACM, 934:1–934:21.

[26] Alessio Ferrari and Paola Spoletini. 2025. Formal Requirements Engineering and Large Language Models: A Two-Way Roadmap. Inf. Softw. Technol. 181, C (April 2025). doi:10.1016/j.infsof.2025.107697

[27] Peter Fettke, Fabiana Fournier, Lior Limonad, Andreas Metzger, Stefanie Rinderle-Ma, and Barbara Weber. 2025. XABPs: Towards eXplainable Autonomous Business Processes. In 4th International Workshop on Process Management in the AI era (PMAI), collocated with the 28th European Conference on Artificial Intelligence (ECAI), Bologna, October 25-30, 2025.

[28] Fabiana Fournier, Lior Limonad, and Yuval David. 2025. Agentic AI Process Observability: Discovering Behavioral Variability. In 4th International Workshop on Process Management in the AI era (PMAI), collocated with ECAI, Bologna, Italy,October 25, 2025.

[29] Iason Gabriel, Geoff Keeling, Arianna Manzini, and James Evans. 2025. We need a new ethics for a world of AI agents. Nature 644 (2025).

[30] Cuiyun Gao, Xing Hu, Shan Gao, Xin Xia, and Zhi Jin. 2025. The Current Challenges of Software Engineering in the Era of Large Language Models. ACM Trans. Softw. Eng. Methodol. 34, 5, Article 127 (May 2025). doi:10.1145/3712005

[31] R. Gowri. 2025. OMACS Based Adaptive Intelligent Tutoring System Development. In 2025 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI). 1–6. doi:10.1109/ICDSAAI65575.2025.11011745

[32] Thomas Grisold, Nicholas Berente, and Stefan Seidel. 2025. Guardrails for Human-AI Ecologies: Norm-Based Coordination and Design for Predictability. Management Information Systems Quarterly (2025).

[33] Alex Gu, Naman Jain, Wen-Ding Li, Manish Shetty, Yijia Shao, Ziyang Li, Diyi Yang, Kevin Ellis, Koushik Sen, and Armando Solar-Lezama. 2025. Challenges and Paths Towards AI for Software Engineering. CoRR abs/2503.22625 (2025). arXiv:2503.22625 doi:10.48550/ARXIV.2503.22625

[34] Everton Guimaraes and Nathalia Nascimento. 2025. AI in the Software Development Lifecycle: Insights and Open Research Questions. In Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25). Association for Computing Machinery, Clarion Hotel Trondheim, Trondheim, Norway and New York, NY, USA, 1353–1357. doi:10.1145/3696630.3730538

[35] Ahmed E. Hassan, Dayi Lin, Gopi Krishnan Rajbahadur, Keheliya Gallaba, Filipe Roseiro Côgo, Boyuan Chen, Haoxiang Zhang, Kishanthan Thangarajah, Gustavo Ansaldi Oliva, Jiahuei (Justina) Lin, Wali Mohammad Abdullah, and Zhen Ming (Jack) Jiang. 2024. Rethinking Software Engineering in the Era of Foundation Models: A Curated Catalogue of Challenges in the Development of Trustworthy FMware. In Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, FSE 2024, Porto de Galinhas, Brazil, July 15-19, 2024, Marcelo d'Amorim (Ed.). ACM, 294–305. doi:10.1145/3663529.3663849

[36] Tobias Hecking, Thorsten Sommer, and Michael Felderer. 2025. An Architecture and Protocol for Decentralized Retrieval Augmented Generation. In ICSA Companion. IEEE, 31–35.

[37] Patrick Hemmer, Max Schemmer, Niklas Kühl, Michael Vössing, and Gerhard Satzger. 2025. Complementarity in human-AI collaboration: concept, sources, and evidence. European Journal of Information Systems 0, 0 (2025), 1–24. doi:10.1080/0960085X.2025.2475962

[38] Alan R. Hevner and Veda C. Storey. 2021. Externalities of Design Science Research: Preparation for Project Success. Vol. 12807 LNCS. Springer International Publishing.

[39] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. ACM Trans. Softw. Eng. Methodol. 33, 8, Article 220 (Dec. 2024). doi:10.1145/3695988

[40] Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. 2024. Self-evolving multi-agent collaboration networks for software development. arXiv preprint arXiv:2410.16946 (2024).
[41] Nicholas R. Jennings. 2000. On agent-based software engineering. Artif. Intell. 117, 2 (2000), 277–296. doi:10.1016/S0004-3702(99)00107-1
[42] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. ACM Comput. Surv. 55, 12 (2023), 248:1–248:38. doi:10.1145/3571730
[43] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A Survey on Large Language Models for Code Generation. arXiv:2406.00515 [cs.CL] https://arxiv.org/abs/2406.00515
[44] Haolin Jin, Linghan Huang, Haipeng Cai, Jun Yan, Bo Li, and Huaming Chen. 2024. From LLMs to LLM-based Agents for Software Engineering: A Survey of Current, Challenges and Future. CoRR abs/2408.02479 (2024). arXiv:2408.02479 doi:10.48550/ARXIV.2408.02479
[45] Wei Jin, Hongzhi Chen, Xiaoyan Wang, Benru Gong, and Xiufeng Lin. 2025. An AI-native application assemble platform for easy-integrating of AIGC based services. In Proceedings of the 2024 4th International Conference on Big Data, Artificial Intelligence and Risk Management (ICBAR '24). Association for Computing Machinery, New York, NY, USA, 578–583. doi:10.1145/3718751.3718843
[46] Hrishikesh Joshi. 2024. Secure Software Architecture for Enterprise Generative Artificial Intelligence. In CSIT. IEEE, 1–5.
[47] Matthew Kam, Cody Miller, Miaoxin Wang, Abey Tidwell, Irene A. Lee, Joyce Malyn-Smith, Beatriz Perret, Vikram Tiwari, Joshua Kenitzer, Andrew Macvean, and Erin Barrar. 2025. What Do Professional Software Developers Need to Know to Succeed in an Age of Artificial Intelligence?. In Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25). Association for Computing Machinery, Clarion Hotel Trondheim, Trondheim, Norway and New York, NY, USA, 947–958. doi:10.1145/3696630.3727251
[48] Jan Keim, Tobias Hey, Vincenzo Scotti, Raffaela Mirandola, and Anne Koziolek. 2025. Towards Automated Knowledge Management in the Software Life Cycle. doi:10.5445/IR/1000181618
[49] Salman Khan and Mendus Daviglus. 2025. AI-Driven Automation in Agile Development: Multi-Agent LLMs for Software Engineering. (2025).
[50] Deepali Kholkar, Suraj Thapa, Akhilesh Pal, and Suman Roychoudhury. 2024. Feature Model-based Integration of Machine Learning in Software Product Lines. In ICSA-C. IEEE, 295–302.
[51] Qichao Kong, Zhengwei Lv, Yiheng Xiong, Dingchun Wang, Jingling Sun, Ting Su, Letao Li, Xu Yang, and Gang Huo. 2025. ProphetAgent: Automatically Synthesizing GUI Tests from Test Cases in Natural Language for Mobile Apps. Association for Computing Machinery, New York, NY, USA, 174–179. https://doi.org/10.1145/3696630.3728543
[52] Alexander Korn, Samuel Gorsch, and Andreas Vogelsang. 2025. LLMREI: Automating Requirements Elicitation Interviews with LLMs. In 33rd IEEE International Requirements Engineering Conference, RE 2025, Valencia, Spain, Sep 1-5, 2025.
[53] Qianlong Lan, Anuj Kaul, Nishant Kumar Das Pattanaik, Piyush Pattanayak, and Vinothini Pandurangan. 2024. Securing Applications of Large Language Models: A Shift-Left Approach. In EIT. IEEE, 1–2.
[54] Seo-young Lee, Matthew V. Law, and Guy Hoffman. 2025. When and How to Use AI in the Design Process? Implications for Human-AI Design Collaboration. Int. J. Hum. Comput. Interact. 41, 2 (2025), 1569–1584. doi:10.1080/10447318.2024.2353451
[55] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. 2025. The Rise of AI Teammates in Software Engineering (SE) 3.0: How Autonomous Coding Agents Are Reshaping Software Engineering. arXiv:2507.15003 [cs.SE] https://arxiv.org/abs/2507.15003
[56] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. 2025. The Rise of AI Teammates in Software Engineering (SE) 3.0: How Autonomous Coding Agents Are Reshaping Software Engineering. CoRR abs/2507.15003 (2025). arXiv:2507.15003 doi:10.48550/ARXIV.2507.15003
[57] Jialong Li, Mingyue Zhang, Nianyu Li, Danny Weyns, Zhi Jin, and Kenji Tei. 2024. Generative AI for Self-Adaptive Systems: State of the Art and Research Roadmap. ACM Trans. Auton. Adapt. Syst. 19, 3 (2024), 13:1–13:60. doi:10.1145/3686803
[58] Jenny T. Liang, Melissa Lin, Nikitha Rao, and Brad A. Myers. 2025. Prompts Are Programs Too! Understanding How Developers Build Software Containing Prompts. Proc. ACM Softw. Eng. 2, FSE (2025), 1591–1614.
[59] Feng Lin, Dong Jae Kim, et al. 2024. Soen-101: Code generation by emulating software process models using large language model agents. arXiv preprint arXiv:2403.15852 (2024).
[60] Qinghua Lu, Liming Zhu, Xiwei Xu, Yue Liu, Zhenchang Xing, and Jon Whittle. 2024. A Taxonomy of Foundation Model based Systems through the Lens of Software Architecture. In CAIN. ACM, 1–6.
[61] Qinghua Lu, Liming Zhu, Xiwei Xu, Zhenchang Xing, Stefan Harrer, and Jon Whittle. 2024. Towards Responsible Generative AI: A Reference Architecture for Designing Foundation Model Based Agents. In ICSA-C. IEEE, 119–126.
[62] Michael R. Lyu, Baishakhi Ray, Abhik Roychoudhury, Shin Hwei Tan, and Patanamon Thongtanunam. 2025. Automatic Programming: Large Language Models and Beyond. ACM Trans. Softw. Eng. Methodol. 34, 5, Article 140 (May 2025). doi:10.1145/3708519
[63] Sanwal Manish. 2024. An autonomous multi-agent llm framework for agile software development. International Journal of Trend in Scientific Research and Development 8, 5 (2024), 892–898.
[64] Antonio Mastropaolo, Camilo Escobar-Velásquez, and Mario Linares-Vásquez. 2025. From Triumph to Uncertainty: The Journey of Software Engineering in the AI Era. ACM Trans. Softw. Eng. Methodol. 34, 5, Article 131 (May 2025). doi:10.1145/3709360
[65] Santiago Matalonga, Domenico Amalfitano, Andrea Doreste, Anna Rita Fasolino, and Guilherme Horta Travassos. 2022. Alternatives for testing of context-aware software systems in non-academic settings: results from a Rapid Review. Information and Software

Technology 149 (2022), 106937. doi:10.1016/j.infsof.2022.106937

[66] Ziran Min and Christof J. Budnik. 2025. Verification and Validation of LLM-RAG for Industrial Automation . In 2025 IEEE International Conference on Artificial Intelligence Testing (AITest). IEEE Computer Society, Los Alamitos, CA, USA, 50–53. doi:10.1109/AITest66680.2025.00012

[67] Lalit Narayan Mishra and Biswaranjan Senapati. 2025. Retail Resilience Engine: An Agentic AI Framework for Building Reliable Retail Systems With Test-Driven Development Approach. IEEE Access 13 (2025), 50226–50243. doi:10.1109/ACCESS.2025.3552592

[68] Nadia Nahar, Christian Kastner, Jenna Butler, Chris Parnin, Thomas Zimmermann., and Christian Bird. 2025. Beyond the Comfort Zone: Emerging Solutions to Overcome Challenges in Integrating LLMs into Software Products . In 2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE Computer Society, Los Alamitos, CA, USA, 516–527. doi:10.1109/ICSE-SEIP66354.2025.00051

[69] Minh Huynh Nguyen, Thang Phan Chau, Phong X. Nguyen, and Nghi D. Q. Bui. 2025. AgileCoder: Dynamic Collaborative Agents for Software Development based on Agile Methodology. In IEEE/ACM Second International Conference on AI Foundation Models and Software Engineering, Forge@ICSE 2025, Ottawa, ON, Canada, April 27-28, 2025. IEEE, 156–167. doi:10.1109/FORGE66646.2025.00026

[70] Anh Nguyen-Duc, Beatriz Cabrero Daniel, Adam Przybylek, Chetan Arora, Dron Khanna, Tomas Herda, Usman Rafiq, Jorge Melegati, Eduardo Guerra, Kai-Kristian Kemell, Mika Saari, Zheying Zhang, Huy Le, Tho Quan, and Pekka Abrahamsson. 2025. Generative Artificial Intelligence for Software Engineering - A Research Agenda. Software Prac. Experience (2025). https://doi.org/10.1002/spe.70005

[71] Ipek Ozkaya, Anita Carleton, John Robert, and Douglas Schmidt. 2023. Application of Large Language Models (LLMs) in Software Engineering: Overblown Hype or Disruptive Change? Carnegie Mellon University, Software Engineering Institute's Insights (blog). https://doi.org/10.58012/6n1p-pw64

[72] Kannan Parthasarathy, Karthik Vaidhyanathan, Rudra Dhar, Venkat Krishnamachari, Adyansh Kakran, Sreemaee Akshathala, Shrikara Arun, Amey Karan, Basil Muhammed, Sumant Dubey, and Mohan Veerubhotla. 2025. Engineering LLM Powered Multi-Agent Framework for Autonomous CloudOps. In 2025 IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN). 201–211. doi:10.1109/CAIN66642.2025.00031

[73] Mauro Pezzè, Matteo Ciniselli, Luca Di Grazia, Niccolò Puccinelli, and Ketai Qiu. 2024. The Trailer of the ACM 2030 Roadmap for Software Engineering. ACM SIGSOFT Softw. Eng. Notes 49, 4 (2024), 31–40. doi:10.1145/3696117.3696126

[74] Mauro Pezzè and Abhik Roychoudhury (Eds.). 2025. ACM Trans. Softw. Eng. Methodol. 34, 5 (2025). https://dl.acm.org/toc/tosem/2025/34/5

[75] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 2023. Chatdev: Communicative agents for software development. arXiv preprint arXiv:2307.07924 (2023).

[76] Ketai Qiu, Niccolò Puccinelli, Matteo Ciniselli, and Luca Di Grazia. 2025. From Today's Code to Tomorrow's Symphony: The AI Transformation of Developer's Routine by 2030. ACM Trans. Softw. Eng. Methodol. 34, 5, Article 121 (May 2025). doi:10.1145/3709353

[77] Zeeshan Rasheed, Malik Abdul Sami, Kai-Kristian Kemell, Muhammad Waseem, Mika Saari, Kari Systä, and Pekka Abrahamsson. 2024. CodePori: Large-Scale System for Autonomous Software Development Using Multi-Agent Technology. arXiv:2402.01411 [cs.SE] https://arxiv.org/abs/2402.01411

[78] Zeeshan Rasheed, Muhammad Waseem, Kai-Kristian Kemell, Wang Xiaofeng, Anh Nguyen Duc, Kari Systä, and Pekka Abrahamsson. 2023. Autonomous Agents in Software Development: A Vision Paper. arXiv:2311.18440 [cs.SE] https://arxiv.org/abs/2311.18440

[79] Sergio Rico and Lena-Maria Öberg. 2025. Challenges and Opportunities for Generative AI in Software Engineering: A Managerial View. In Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25). Association for Computing Machinery, Clarion Hotel Trondheim, Trondheim, Norway and New York, NY, USA, 1338–1344. doi:10.1145/3696630.3728718

[80] Krishna Ronanki. 2025. Facilitating Trustworthy Human-Agent Collaboration in LLM-based Multi-Agent System oriented Software Engineering. Association for Computing Machinery, New York, NY, USA, 1333–1337. https://doi.org/10.1145/3696630.3728717

[81] Pat Rondon, Renyao Wei, José Cambronero, Jürgen Cito, Aaron Sun, Siddhant Sanyam, Michele Tufano, and Satish Chandra. 2025. Evaluating agent-based program repair at google. arXiv preprint arXiv:2501.07531 (2025).

[82] Malik Abdul Sami, Muhammad Waseem, Zeeshan Rasheed, Mika Saari, Kari Systä, and Pekka Abrahamsson. 2024. Experimenting with multi-agent software development: Towards a unified platform. arXiv preprint arXiv:2406.05381 (2024).

[83] Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. 2025. AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges. CoRR abs/2505.10468 (2025). arXiv:2505.10468 doi:10.48550/ARXIV.2505.10468

[84] Jaakko Sauvola, Sasu Tarkoma, Mika Klemettinen, Jukka Riekki, and David Doermann. 2024. Future of Software Development with Generative AI. Automated Software Engg. 31, 1 (March 2024). doi:10.1007/s10515-024-00426-z

[85] Yuchen Shao, Yuheng Huang, Jiawei Shen, Lei Ma, Ting Su, and Chengcheng Wan. 2025. Are LLMs Correctly Integrated into Software Systems?. In ICSE. IEEE, 1178–1190.

[86] Hui Song, Arda Goknil, Xiaojun Jiang, Espen Melum, Hyunwhan Joe, Caterina Gazzotti, Valerio Frascolla, Adela Nedisan Videsjorden, and Phu Nguyen. 2025. Developing Multi-Agent LLM Applications Through Continuous Human-LLM Co-Programming. In 2025

IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN). 42–47. doi:10.1109/CAIN66642.2025.00013

[87] Margaret-Anne Storey, Daniel Russo, Nicole Novielli, Takashi Kobayashi, and Dong Wang. 2024. A Disruptive Research Playbook for Studying Disruptive Innovations. ACM Trans. Softw. Eng. Methodol. 33, 8, Article 195 (Nov. 2024). doi:10.1145/3678172

[88] Valerio Terragni, Partha S. Roop, and Kelly Blincoe. 2024. The Future of Software Engineering in an AI-Driven World. CoRR abs/2406.07737 (2024). arXiv:2406.07737 doi:10.48550/ARXIV.2406.07737

[89] Valerio Terragni, Annie Vella, Partha Roop, and Kelly Blincoe. 2025. The Future of AI-driven Software Engineering. ACM Trans. Softw. Eng. Methodol. 34, 5, Article 120 (May 2025). doi:10.1145/3715003

[90] Bianca Trinkenreich, Fabio Calefato, Geir Hanssen, Kelly Blincoe, Marcos Kalinowski, Mauro Pezzè, Paolo Tell, and Margaret-Anne D. Storey. 2025. Get on the Train or be Left on the Station: Using LLMs for Software Engineering Research. CoRR abs/2506.12691 (2025). arXiv:2506.12691 doi:10.48550/ARXIV.2506.12691

[91] Hoang Vu, Nataliia Klievtsova, Henrik Leopold, Stefanie Rinderle-Ma, and Timotheus Kampik. 2025. Agentic Business Process Management: Practitioner Perspectives on Agent Governance in Business Processes. In Responsible BPM Forum, 23rd International Conference Business Process Management (BPM), Sevilla, Spain, September 1-5, 2025, Proceedings (Lecture Notes in Computer Science). Springer.

[92] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2024. Software Testing With Large Language Models: Survey, Landscape, and Vision. IEEE Trans. Software Eng. 50, 4 (2024), 911–936. doi:10.1109/TSE.2024.3368208

[93] Tianjia Wang, Ramaraja Ramanujan, Yi Lu, Chenyu Mao, Yan Chen, and Chris Brown. 2024. DevCoach: Supporting Students in Learning the Software Development Life Cycle at Scale with Generative Agents. In Proceedings of the Eleventh ACM Conference on Learning @ Scale (Atlanta, GA, USA) (L@S '24). Association for Computing Machinery, New York, NY, USA, 351–355. doi:10.1145/3657604.3664663

[94] Mubeen Wasif and David Tunkel. 2025. Multi-Agent Collaboration in AI: Enhancing Software Development with Autonomous LLMs. doi:10.13140/RG.2.2.31588.08328

[95] Irene Weber. 2024. Large Language Models as Software Components: A Taxonomy for LLM-Integrated Applications. arXiv:2406.10300 [cs.SE] https://arxiv.org/abs/2406.10300

[96] Zhengyuan Wei, Albert T.L. Lee, Victor C.S. Lee, and Wing-Kwong Chan. 2024. Toward AI-facilitated Learning Cycle in Integration Course Through Pair Programming with AI Agents. In 2024 36th International Conference on Software Engineering Education and Training (CSEE&T). 1–5. doi:10.1109/CSEET62301.2024.10663037

[97] Roel Wieringa. 2009. Design science as nested problem solving. In Proceedings of the 4th international conference on design science research in information systems and technology.

[98] Michael Wooldridge. 2009. An introduction to multiagent systems (second ed.). John wiley & sons.

[99] Michael Wooldridge, Nicholas R Jennings, and David Kinny. 1999. A methodology for agent-oriented analysis and design. In Proceedings of the third annual conference on Autonomous Agents. 69–76.

[100] Michael J. Wooldridge and Paolo Ciancarini. 2000. Agent-Oriented Software Engineering: The State of the Art. In Agent-Oriented Software Engineering, First International Workshop, AOSE 2000, Limerick, Ireland, June 10, 2000, Revised Papers (Lecture Notes in Computer Science, Vol. 1957), Paolo Ciancarini and Michael J. Wooldridge (Eds.). Springer, 1–28. doi:10.1007/3-540-44564-1_1

[101] Yin Wu, Haijun Wang, Yuanhui Zhang, Xitao Li, Hao Wu, Ming Fan, and Ting Liu. 2024. Business Compliance Detection of Smart Contracts in Electricity and Carbon Trading Scenarios. In 35th IEEE International Symposium on Software Reliability Engineering, ISSRE 2024 - Workshops, Tsukuba, Japan, October 28-31, 2024. IEEE, 177–178. doi:10.1109/ISSREW63542.2024.00074

[102] Da Xu, Danqing Zhang, Chuanwei Ruan, Lingling Zheng, Bo Yang, Guangyu Yang, Shuyuan Xu, and Haixun Wang. 2025. Tutorial on Landing Generative AI in Industrial Social and E-commerce Recsys. In WWW (Companion Volume). ACM, 57–60.

[103] Yisen Xu. 2025. MUARF: Leveraging Multi-Agent Workflows for Automated Code Refactoring. In 2025 IEEE/ACM 47th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). 226–227. doi:10.1109/ICSE-Companion66252.2025.00071

[104] Quanjun Zhang, Chunrong Fang, Yuxiang Ma, Weisong Sun, and Zhenyu Chen. 2024. A Survey of Learning-based Automated Program Repair. ACM Trans. Softw. Eng. Methodol. 33, 2 (2024), 55:1–55:69. doi:10.1145/3631974

[105] Yiran Zhang, Ruiyin Li, Peng Liang, Weisong Sun, and Yang Liu. 2025. Knowledge-Based Multi-Agent Framework for Automated Software Architecture Design. In Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (Clarion Hotel Trondheim, Trondheim, Norway) (FSE Companion '25). Association for Computing Machinery, New York, NY, USA, 530–534. doi:10.1145/3696630.3728493

[106] Wenting Zhao, Nan Jiang, Celine Lee, Justin T Chiu, Claire Cardie, Matthias Gallé, and Alexander M Rush. 2024. Commit0: Library generation from scratch. arXiv preprint arXiv:2412.01769 (2024).

A Search strings used for literature review

A.1 Search Strings for GenAI Copilots

Search String ACM

”query”: { Fulltext:((”software development life cycle” OR ”SDLC” OR ”life*cycle” OR ”development process” OR ”software engineering process”) AND (”LLM?” OR ”Large Language Model?” OR ”GenAI” OR ”Generative AI” OR ”Generative Artificial Intelligence” OR ”transformer architecture”)) AND ContentGroupTitle:(”ICSE:” OR ”ASE:” OR ”FSE:” OR ”ACM Transactions on Software Engineering and Methodology” OR ”IEEE Transactions on Software Engineering” OR ”Empirical Software Engineering” OR ”Journal of Systems and Software” OR ”Information and Software Technology” OR ”Automated Software Engineering”) } ”filter”: { E-Publication Date: (01/01/2019 TO *) }

Search String IEEE

((”Full Text Only”: ”software development life cycle”) OR (”Full Text Only”: ”SDLC”) OR (”Full Text Only”:”life*cycle”) OR (”Full Text Only”:”development process”) OR (”Full Text Only”:”software engineering process”)) AND ((”Full Text Only”: ”LLM?”) OR (”Full Text Only”:”Large Language Model?”) OR (”Full Text Only”:”GenAI”) OR (”Full Text Only”:”Generative AI”) OR (”Full Text Only”:”Generative Artificial IntelligenceI”) OR (”Full Text Only”:”transformer architecture”)) AND ((”Publication Title”: ”*International Conference on Software Engineering *ICSE”) OR (”Publication Title”: ”*Automated Software Engineering *ASE”) OR (”Publication Title”: ”*Foundations of Software Engineering”) OR (”Publication Title”: ”IEEE Transactions on Software Engineering”) OR (”Publication Title”: ”Empirical Software Engineering”) OR (”Publication Title”: ”Journal of Systems and Software”)) AND ((”Publication Year”: ”2019”) OR (”Publication Year”: ”2020”) OR (”Publication Year”: ”2021”) OR (”Publication Year”: ”2022”) OR (”Publication Year”: ”2023”) OR (”Publication Year”: ”2024”) OR (”Publication Year”: ”2025”))

A.2 Search strings for GenAI Teammate

Search String Google Scholar/Scopus

ABS (Agent software development generative AI) OR ABS (Agentic software development generative AI) OR ABS (Agent software development lifecycle generative AI) OR ABS (Agentic software development lifecycle generative AI) OR ABS (Agent sdlc generative AI) OR ABS (Agentic sdlc generative AI) OR ABS (Agent software development genAI) OR ABS (Agentic software development genAI) OR ABS (Agent software development lifecycle genAI) OR ABS (Agentic software development lifecycle genAI) OR ABS (Agent sdlc genAI) OR ABS (Agentic sdlc genAI)

A.3 Search strings for GenAIware

Search String ACM/IEEE/Scopus
("software development life?cycle" OR "sdlc" OR "software development process" OR "software engineering process" OR "development process" OR "software system" OR "software application" OR "software architecture" OR "system design" OR "application development" OR "software component" OR "software service" OR "system development" OR "application design" OR "application" OR "software component" OR "software product") AND ("llm?" OR "large language model?" OR "fine?tuned llm?" OR "fine?tuned language model?" OR "fine?tuned lm?" OR "fine?tuned language model?" OR "gen?ai" OR "generative?ai" OR "generative artificial intelligence" OR "transformer model" OR "transformer language model" OR "transformer lm" OR "neural language model" OR "neural lm" OR "instruction tuning" OR "instruction fine?tuning" OR "instruction following" OR "retrieval?augmented generation" OR "rag" OR "assistant tuning"OR "assistant fine?tuning" OR "function calling" OR "tool use" OR "guardrails") AND ("use case" OR "functionalit*" OR "integrat*" OR "deploy*" OR "architectur*" OR "backend" OR "micro?service" OR "application" OR "orchestration" OR "agent orchestration" OR "safety filter" OR "filter" OR "pipeline" OR "workflow" OR "service" OR "framework") AND ("ai?ware" OR "gen?ai?ware" OR "semantic search" OR "gopher" OR "agent orchestration" OR "multi?agent orchestration" OR "inference engine" OR "dev?ops 2.0" OR "prompt?engineering" OR "prompt?ware")

A.4 Search string for GenAI Robot

Search String Scopus
("agentic" OR "agent-based") AND ("LLM" OR "large language model" OR "generative AI" OR "Gen AI" OR "transformer" OR "GPT" or "BERT" OR "Generative Adversarial Network" OR "GANs" or "Variational Autoencoder" OR "VAE") AND ("applications" OR "use cases" OR "impacts" OR "effects") AND ("software" OR "software development" OR "software engineering") AND PUBYEAR > 2023 AND PUBYEAR < 2027 AND (LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp")) AND (LIMIT-TO (LANGUAGE , "English"))

B Prompts used for literature review

B.1 GenAI Copilot

Prompt invoked for Qwen3 version qwen3:8b_q4_K_M (commit 500a1f067a9f)[16] using Ollama.
System prompt:

```
You are a helpful academic assistant. If you include reasoning, enclose it in <
    think> tags and give a clear 'Final Answer:' at the end.
```

User message Q1:

```
Title: {title}
Abstract: {abstract}
Keywords: {keywords}
Question: Based on title, abstract and keywords provided is there a chance that the
     paper discusses the role and/or implications of GenAI or LLMs on the software
    development process or lifecycle? Answer only with yes, no, or maybe.
```

[16]model card: https://ollama.com/library/qwen3:8b

User message Q2:

```
Title: {title}
Abstract: {abstract}
Keywords: {keywords}
Question: Based on title, abstract and keywords provided does the paper utilize
    GenAI or LLMs as a tool to support or automate tasks along the software
    development process or lifecycle (and not as part of the final software system,
    product, or application)? Answer only with yes or no.
```

B.2 GenAI Teammate

In the context of GenAI Teammate, no AI model was involved in the analysis of the paper during the Rapid Literature Review.

B.3 GenAIware

Prompt invoked for LLM Qwen3 version unsloth/Qwen3-4B-Instruct-2507-GGUF[17] using the Llama CPP python library version 0.3.16.

System prompt:

```
The following is a conversation between a human user and an AI assistant. The
    assistant is knowledgeable in several research areas including computer science,
     software engineering and artificial intelligence. In this dialogue, the
    assistant answers to the user's questions and requests straightforwardly and
    concisely, providing detailed explanations only when required.
```

User message Q1:

```
Based on the following paper metadata (title, abstract, and keywords), assess
    whether this paper likely contains relevant information about using Generative
    AI (e.g., Large Language Models) as a component or a service of a software
    system (and not as a development tool), specifically from a software development
     lifecycle perspective.
Answer yes or no depending on whether the paper discusses ways to augment the
    software development life cycle when using Generative AI to realise software
    functionalities that would otherwise not be available, then explain your
    response.

Title: "{TITLE}"

Abstract:

{ABSTRACT}

Keywords: {KEYWORDS}
```

User message Q2:

[17] model card: https://huggingface.co/unsloth/Qwen3-4B-Instruct-2507-GGUF

```
Based on the following paper metadata (title, abstract, and keywords), assess
    whether this paper likely contains relevant information about the design,
    implementation, or architectural considerations deriving from the use of
    Generative AI (e.g., Large Language Models) as a component or a service of a
    software system, specifically from a software development lifecycle perspective.
Answer yes or no depending on whether the paper discusses ways to augment the
    software development life cycle when using Generative AI to realise software
    functionalities that would otherwise not be available, then explain your
    response.

Title: "{TITLE}"

Abstract:

{ABSTRACT}

Keywords: {KEYWORDS}
```

B.4 Prompts for GenAI Robots

Prompt invoked for LLM version gemini-2.5-flash via Python code using the Google GenAI Python library version 1.24.0.

```
<Role> You are a researcher of software engineering that analyses research papers
    found via a literature data base search.

<Context> The data provides the title, abstract and keywords of each research paper
     that is supposed to cover argentic AI as part of software systems and
    applications.

Agentic AI means that part of the system is realized by an AI system or model,
    which (a) operates with a greater degree of autonomy, (b) is capable of
    undertaking (human) roles, (c) manages multi-step tasks, (d) proactively
    collaborates with human developers or other agents, (c) achieves higher-level
    goals.

<Task> Your task is to analyze the paper and provide an answer to the following
    research question.

<Output> Answer this question by providing your answers as bullet points. Use the
    character '*' as bullets. Only provide the answers, but nothing else, i.e., do
    not repeat the given question, your train of thought, or additional background
    information.

Before providing the bullet-style answer to the question, give a general assessment
     of whether the question can be answered with TRUE or FALSE.
```

```
<Question 1> Does the paper satisfy the following criteria?
(a) The paper utilizes Agentic AI as part of the final software system, product, or
     application.
(b) The paper does NOT only focus on using Agentic AI as part of the software
    development process.
If both criteria are met, answer with TRUE.
If at least one of the criteria is NOT met, answer with FALSE.

<Question 2> Does the paper discuss the role and/or implications of Agentic AI on
    the software development process or lifecycle (such as CI/CD, DevOps, Agile
    Development, Scrum, XP, etc.)?
```