# Challenges and Opportunities for Generative AI in Software Engineering: A Managerial View

Sergio Rico
Lena-Maria Öberg
sergio.rico@miun.se
lena-maria.oberg@miun.se
Department of Communication, Quality Management and Information Systems
Mid Sweden University
Östersund, Sweden

## Abstract

Generative artificial intelligence (GenAI) has shown a strong potential to automate tasks, increase productivity, and enhance software quality. Managers have an important role in adopting GenAI as they lead strategic planning and team coordination. This study examines the software engineering tasks that managers consider suitable for large language models (LLMs), as well as the role of management and education in software engineering practices. The study reviews current applications in practice and explores the challenges encountered during implementation. We conducted a workshop with managers from diverse organizations to collect insights on the growing use of LLMs in areas such as code development, debugging, documentation, and training. The discussion highlighted several key concerns, including intellectual property issues, quality assurance, and integration difficulties. Overall, our findings suggest that LLMs can potentially transform software engineering practices if technical and organizational challenges are carefully addressed.

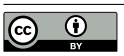
## CCS Concepts

• **Social and professional topics** → **Software management**; • **Software and its engineering** → *Software development process management*; **Software development techniques**.

## Keywords

Generative AI (GenAI), Large Language Models (LLMs), Software Engineering, Adoption, Empirical Study, Code Development, Software Quality Assurance

**ACM Reference Format:**
Sergio Rico and Lena-Maria Öberg. 2025. Challenges and Opportunities for Generative AI in Software Engineering: A Managerial View. In *33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25), June 23–28, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3696630.3728718



This work is licensed under a Creative Commons Attribution 4.0 International License.
*FSE Companion '25, Trondheim, Norway*
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1276-0/2025/06
https://doi.org/10.1145/3696630.3728718

## 1 Introduction

Generative AI (GenAI) and particularly large language models (LLMs) have attracted significant attention due to their ability to process and generate text with remarkable fluency and coherence [1, 2]. These technological developments offer new opportunities to automate tasks, boost productivity, and enhance software quality [3–5]. As GenAI continues to evolve rapidly, it raises important questions about how these improvements may reshape software development practices and education [6].

As highlighted by Zheng et al. [7], who surveyed 123 recent papers integrating LLMs in software engineering tasks, there is a wide range of software engineering activities where LLMs could have a high impact, such as requirements, design, testing, deployment, and maintenance. While many existing studies explore the technical capabilities of LLMs or investigate how individual practitioners use these tools [5], a growing body of work shows that organizational and managerial perspectives play an equally critical role [5, 8–10]. Managers, in particular, play a key role in technology adoption by setting strategic priorities, allocating resources, and guiding software teams. These elements are key to integrating GenAI capabilities into software engineering workflows.

GenAI's role may extend far beyond code generation as software engineering practices evolve, potentially reshaping the entire software development life cycle [11]. Researchers and practitioners are calling for investigations that span requirements, design, testing, deployment, and maintenance. They advocate for a better understanding of how GenAI-driven tools can be effectively integrated [5, 9, 12–14]. This study contributes to ongoing life cycle-oriented discussions and complements prior work by emphasizing managerial viewpoints that shape organizational readiness.

In our study, we conducted a workshop with managers from various organizations to gather their insights on using LLM in key software engineering tasks. **Our contributions are twofold:**

(1) **Analyzing Potential and Actual Uses of LLMs:** We compare potential applications identified in the literature with actual implementations reported by managers.
(2) **Identifying Practical Challenges:** We examine managerial perspectives on challenges associated with LLM adoption, such as intellectual property issues, quality concerns, and integration difficulties.

## 2 Background and Related Work

Recent advances in GenAI, particularly LLMs, have opened new pathways to automate and enhance tasks in software engineering, such as code completion, test generation, and artifact creation [4, 15]. An increasing number of studies explore potential applications of LLMs for software development, ranging from synthesizing code and refactoring to supporting various testing and maintenance activities [1, 3]. However, despite these promising directions, few studies focus on how organizations adopt GenAI in real-world settings.

Some recent studies have begun addressing this gap by incorporating empirical data on LLM usage. For example, Coutinho et al. [16] conducted a case study with software professionals to assess how GenAI tools affect productivity, while Tufano et al. [17] systematically analyzed over 1,500 GitHub instances that mention ChatGPT to form a taxonomy of automated tasks. Khojah et al. [18] studied professional software engineers using ChatGPT over the course of a week. Their study offers insights into trust factors, usage patterns, and the perceived benefits of LLMs. Complementary industry surveys, such as those reported by Jahić et al. [12], further show that LLMs are being adopted for tasks like code assistance and generation. However, their integration often requires significant human oversight. There are also several studies focusing on adoption and readiness for generative AI, either at the individual or organizational level. Russo [10] conducted a mixed-methods study, focusing on individual adaptation and the factors that influence it. Another example is the study by Ulfsnes et al. [8] which, based on 13 qualitative interviews, presents what drives individuals to adopt GenAI. Finally, Prasad et al. [9] present key organizational adoption factors. Their study employs a quantitative approach, encompassing 108 companies in India. These companies are not only software companies. Based on this, we argue that studies are needed that focus on the managerial view and, thereby, the organizational level, as existing studies are few and do not adequately address software engineering.

Despite these promising directions, multiple challenges remain underexamined. Sallou et al. [19] draw attention to privacy risks and the black-box nature of closed-source LLMs, where unannounced provider changes can impact results replicability. Intellectual property (IP) issues also arise when LLMs potentially reproduce copyrighted code or confidential data, creating uncertainties about legal compliance. Moreover, the hallucination phenomenon in which LLMs produce fluent but incorrect or off-topic information compromises reliability [4]. Addressing these challenges demands greater transparency, version control, rigorous evaluation strategies for both open- and closed-source models, and stronger safeguards to mitigate data leakage and privacy concerns.

While prior studies highlight GenAI's technical strengths and limitations, they offer limited insight into the organizational and managerial aspects of integrating GenAI into software engineering practices.

## 3 Research Method

We conducted an exploratory study to understand which software engineering tasks managers consider suitable for use with LLMs, how they are currently used, and what challenges arise in practice. While our broader research project also addresses organizational readiness and strategic alignment, this paper focuses on the following research question:

**RQ:** *Which software engineering tasks do managers believe LLMs can support, which tasks are they currently applying LLMs to, and what challenges do they face in doing so?*

We followed three main steps to address this question:

(1) **Literature Review:** We searched digital libraries (ACM, IEEE, and arXiv) for papers published since 2023 using terms such as "LLM," "generative AI," and "software engineering." Instead of a full systematic review, we focused on well-cited secondary studies that classify LLM-related tasks (e.g., [4, 15]) and exemplify technical advances and real-world adoption. This review informed our selection of LLM-supported tasks for data collection.

(2) **Workshop with Managers:** We organized a one-day workshop in April 2024, inviting eight managers and team leaders from diverse organizational backgrounds. The participants represented various roles, industries, and company sizes, as summarized in Table 1. We sent invitations to companies within our network with whom we frequently collaborate in education. These companies are located in or have offices in our city and are based in the Nordics. Although the companies are based in the Nordics, most of them operate with employees at multiple locations in Sweden and other parts of Europe. They also have customers throughout Europe. The selection of companies is a convenience sample, consisting of organizations that are good representatives for software engineering tasks.

| Participant Role | Industry | Company Size |
|---|---|---|
| Compliance Manager | IT Services | Large |
| Product Owner | Public Sector | Large |
| Competence Development Unit Member | - | Small |
| Consulting Manager | IT Consulting | Large |
| Risk and Capital Analysis Manager | Financial Services | Large |
| Team Leader | IT Services | Large |
| Project Leader | Regional Development | Small |
| Senior System Developer | Software Development | Medium |

**Table 1: Summary of Workshop Participants**

The workshop consisted of two segments led by the co-authors and supported by two facilitators who took notes and ensured the smooth flow of the workshop. The first segment focused on organizational AI readiness, while the second segment, which is the focus of this paper, examined LLM usage in software engineering. In this session, participants used colored Post-it cards to indicate:

- **Yellow cards:** Potential uses of LLMs in their projects or departments.
- **Green cards:** Current, real-world applications of LLMs.
- **Red cards:** Major barriers or risks (e.g., IP issues, security, maintainability) complicating LLM adoption.

Following a one-hour open discussion structured along the guidelines by Thoring et al. [20], facilitators helped cluster related ideas while ensuring participant anonymity. All notes and card responses were subsequently digitized and aggregated for analysis.

(3) **Analysis and Reporting:** Two researchers independently conducted a thematic synthesis of the workshop data. The colored Post-it cards were pre-sorted as participants placed them in designated areas on the wall. These cards were merged, summarized, and connected with discussion notes from the LLM adoption segment and the interactive session. Initial coding was compared, and discrepancies were discussed until consensus was achieved, ensuring reliability. This combination allowed for identifying common themes related to LLM usage and its challenges.

## 4 Results

This section presents how managers perceive using LLMs across different software engineering tasks, highlighting both practical applications and prominent challenges. Table 2 provides an overview of the main tasks discussed, the degree of LLM adoption, and key emerging barriers.

### 4.1 Requirements and Design

Although the literature highlights the potential of LLMs to improve clarity and consistency in software requirements [21], most managers in our workshop viewed LLM-driven *requirements generation* and *early design proposals* as mostly aspirational. A few participants reported sporadic, pilot-level usage, citing concerns that ambiguous or verbose outputs might exacerbate stakeholder misunderstandings. One public sector manager indicated they had "toyed with LLM-based design suggestions" but faced organizational skepticism when non-coding tasks relied on AI.

*Current Uses.* So far, participants described only *limited or pilot usage* of LLMs for requirements or design activities. Most attempts involved internal prototypes, exploratory trials, and informal experiments rather than fully integrated practices. Managers noted that although these text-heavy tasks appear theoretically well-suited for LLMs, actual adoption remains minimal.

*Potential Uses.* Managers acknowledged that LLMs could *generate or refine requirements* and produce *automated design proposals*, streamlining early project phases. They viewed these capabilities as helpful for brainstorming or quickly assembling draft specifications—particularly in contexts where documentation is voluminous or time-constrained.

*Challenges.* Despite recognizing the potential, managers reported *organizational hesitation for non-coding tasks*, worried that ambiguous outputs might compound stakeholder misunderstandings. This caution aligns with a general reluctance to trust GenAI in higher-level tasks requiring extensive domain expertise and multi-stakeholder input.

### 4.2 Code Development and Maintenance

Code development tasks emerged as the most mature area of LLM adoption. Multiple participants already employ ChatGPT or GitHub Copilot tools to generate code snippets, provide quick bug fixes, and document functions. According to one consulting manager in the workshop, Copilot suggested 60–70% of routine code, freeing developers to focus on complex features.

*Current Uses.* Some managers reported *interest and active use* of ChatGPT or Copilot to assist with code creation, quick refactoring, and documentation generation. This assistance has already become part of daily workflows for some teams, reducing the burden of repetitive coding tasks. Some other managers mentioned being aware of the potential of LLMs for code development but had not yet implemented them in their teams.

*Potential Uses.* Participants expressed interest in *automated debugging* and *code optimization*, in line with emerging literature on intelligent bug localization and performance tuning [15]. Some see real-time code analysis as a "game-changer," though they stress that human oversight remains crucial to catch subtle or domain-specific errors.

*Challenges. Intellectual property (IP) concerns* were frequently raised specifically in the context of code generation. Managers worried about inadvertently pulling in third-party code or violating licensing terms. Another challenge was ensuring the quality and security of the generated code itself. LLM-generated snippets work for common cases, but participants noted silent failures in edge scenarios. Moreover, specific questions about the maintainability of AI-generated code arose, focusing on the difficulty future developers might face in understanding or modifying code with uncertain origins or design rationale.

### 4.3 Testing and Quality Assurance

Conversations about testing and QA reflected strong interest in *synthetic test data creation* and *test case generation*, but most usage has involved internal prototypes or small-scale pilot tools, which have not been integrated into production environments. One IT services manager described a pilot project that employed LLMs to generate end-to-end tests for a web application. However, they continue to manually re-check each GenAI-generated test to ensure correctness.

*Current Uses.* Participants have conducted *limited internal trials using LLM-based QA tools*, such as generating test cases or synthetic data, typically within sandbox environments. These efforts remain exploratory and have not been deployed in live development pipelines. Nonetheless, managers see the potential for partial automation to accelerate test creation and expand coverage.

*Potential Uses. Automated test generation* and *synthetic data creation* were the top new use cases. Literature similarly points to the advantages of realistic, on-demand test data [17, 22]. Managers believe that proven reliability in these areas could reduce QA workloads while catching more defects early.

*Challenges.* The primary concerns were ensuring the correctness of test outputs and overcoming a high trust threshold for QA tasks. Participants did not elaborate on specific metrics or evaluation criteria being used for the generated tests, reinforcing the experimental nature of current adoption and the reliance on manual validation for quality assurance.

### 4.4 Management and Education

Only a few participants reported experimenting with LLMs for software management tasks such as GenAI-driven effort estimation

**Table 2: Summary of Workshop Insights: Key Tasks, Current/Potential Uses, and Challenges**

| Task Area | Potential Uses | Current Uses | Key Challenges |
|---|---|---|---|
| **Requirements & Design** | Generating or refining requirements, Automated design proposals | Limited or pilot usage only | Risk of ambiguous outputs, Organizational hesitation for non-coding tasks |
| **Code Dev. & Maintenance** | Real-time code suggestions, Debugging, optimization | Tools like ChatGPT, Copilot actively in use, Automatic documentation in some teams | Intellectual property (IP) concerns, Quality/security of GenAI-generated code, Ongoing maintainability issues |
| **Testing & QA** | Automated test generation, Synthetic data creation | Mostly internal trials or prototype evaluations | Ensuring correctness of test outputs, High trust threshold for QA tasks |
| **Management & Education** | AI-driven effort estimation, Internal training, prompt engineering | Skill-building workshops, Limited pilot use for estimations | Uncertainty over long-term GenAI impact, Data privacy for training materials |

or project planning. More frequently, managers mentioned the role of LLMs in education and training, particularly generating tutorials or supporting "prompt engineering" clinics for developers.

Beyond purely technical considerations, the interpersonal dynamics of working with code-suggesting LLMs have drawn increasing attention. For instance, Barke et al. [23] explored how developers engage with Copilot-like models, emphasizing the need for collaborative strategies and frequent validation to maintain trust in GenAI-driven code suggestions. Our workshop participants similarly pointed out the importance of iterative feedback loops when integrating LLM-based tools into existing workflows, especially for training developers new to GenAI features.

*Current Uses.* Some organizations use LLM-produced tutorials to onboard junior developers or upskill existing staff. Skill-building workshops often involve short sessions on effective prompting. One financial services manager highlighted how LLM-generated materials speed up internal knowledge-sharing.

*Potential Uses.* Managers saw potential in AI-driven effort estimation, forecasting, and broader project-planning features if reliability and model drift issues could be addressed. They also recognized that next-generation LLMs might offer more advanced tutoring capabilities or real-time coaching.

*Challenges.* Uncertainty over long-term GenAI impact hindered investment in LLM-based management tools, as managers struggled with validating the outputs. Additionally, data privacy for training materials remains a concern—particularly in regulated industries where managers worry about exposing sensitive information or inadvertently training public models with proprietary data.

### 4.5 Summary of Findings

Overall, managers reported the most tangible LLM usage in coding tasks, followed by early pilots in testing and educational content development. Requirements engineering and design activities remain more exploratory, as organizations wrestle with trust and interpretability issues. While the potential for LLMs to transform software engineering is widely acknowledged, concerns over IP, security, and maintainability support the need for additional guardrails and best practices.

## 5 Discussion

Our findings show how LLMs are beginning to transform traditional software engineering tasks, from code development and testing to maintenance and education. We observe that:

(1) **LLMs can streamline routine tasks:** Managers reported tangible benefits of LLMs for generating code snippets, accelerating bug fixes, and offering real-time suggestions. These uses hint at a near future where daily development tasks are partially offloaded to AI-powered assistants, potentially reshaping how teams allocate time and expertise.

(2) **IP and security risks are major cross-cutting barriers:** Beyond specific coding tasks, concerns about IP infringement and data security emerged as significant organizational hurdles. Managers emphasized the importance of clearer legal frameworks and guarantees of data privacy before broader adoption could occur. The lack of transparency regarding training data and potential data leakage remains a key inhibitor.

(3) **Maintainability and trust issues hinder adoption throughout the SDLC:** Similarly, concerns about the long-term maintainability of AI-generated artifacts and the overall trustworthiness of LLM outputs represent significant, cross-cutting hurdles.

While tools show promise, the need for constant human validation, especially for complex or critical tasks, limits their current impact and scalability. This reflects broader challenges in establishing reliable and verifiable AI systems within engineering workflows.

(4) **Adoption remains cautious:** Although enthusiasm for code-centric applications is high, tasks involving requirements engineering and design remain exploratory. Testing efforts often take the form of sandboxed trials or prototypes rather than full production deployments. These patterns reflect hesitations around the reliability of AI outputs for complex or domain-specific tasks, indicating a need for more rigorous validation.

(5) **Organizational factors significantly influence GenAI adoption:** Intellectual property and security issues emerged as central concerns, suggesting that end-to-end workflows will require robust data governance, licensing strategies, and collaboration across legal, development, and security teams. Moreover, managers expressed uncertainty about integrating AI-generated code into long-term maintenance workflows.

(6) **Future data collection is key for understanding GenAI's broader impact:** To evaluate LLMs in the software lifecycle context, researchers and practitioners must gather empirical data not only on short-term productivity gains but also on code quality, maintainability, and organizational adoption over time. Notably, we conducted this workshop in April 2024, and we plan to follow up with the participating managers one year later to assess how the rapidly evolving GenAI4SE has transformed organizational practices and outcomes.

## 5.1 Education for AI-Augmented Software

Our results indicate that while LLMs can streamline software engineering tasks, education remains critical for effective adoption. Given our dual role as researchers and educators, we recognize that training and continuous learning initiatives are essential to bridge the gap between theoretical GenAI capabilities and practical, ethically grounded software engineering practices. Several workshop participants emphasized the importance of training and guidance for developers, especially regarding prompt engineering, data governance, and the reliability of AI outputs. Managers described running internal workshops and GenAI "clinics" to build confidence and competence in using LLMs.

From a teaching perspective, these findings suggest that software engineering education must increasingly integrate core topics of the AI lifecycle, such as model training, bias detection, and responsible deployment. This approach can help practitioners and managers better navigate issues like model drift, hallucinations, and intellectual property concerns. Regarding curriculum design, we emphasize the need to include clear competencies that prepare students for environments where GenAI is actively used. These include not only technical skills, such as understanding the lifecycle of artificial intelligence systems, prompt engineering, and evaluating model outputs, but also broader skills like ethical reasoning, awareness of data practices, and the ability to critically assess the use of these models in development. Including these topics in core and elective courses would help prepare graduates to engage with the opportunities and challenges of GenAI in practice.

LLMs also offer novel learning opportunities by acting as interactive tutors or content generators for onboarding developers. However, participants noted risks such as outdated model knowledge and inadvertently reinforcing incorrect code patterns. To address these concerns, educators and organizational leaders should balance convenience with robust validation measures. Developers must critically evaluate AI-generated outputs rather than accepting them at face value.

In sum, upskilling remains relevant as AI-augmented software becomes more prevalent. GenAI will bring transformative benefits only if teams have the necessary skills and judgment to deploy them responsibly. Bridging GenAI's theoretical capabilities with practical, ethically grounded software engineering practices requires close collaboration between industry, educators, and researchers to shape comprehensive training programs and continuous learning initiatives.

## 5.2 Social Influence on LLM Adoption

An interesting dimension of LLM adoption is the extent to which individual developers act independently versus relying on organizational or peer guidance. Recent work by Russo [5] found that 29% of respondents experienced no influence from colleagues or peers when deciding to adopt LLMs. These developers reported relying on their own assessments and self-efficacy rather than external opinions. In contrast, other participants in Russo's study reported moderate or even high social influence, showing that social-cognitive factors can vary widely in software engineering teams.

Our workshop data indicates a tension between this independent decision-making and the managers' view that organizational policies or guidelines are essential. While some individual developers in the participating companies were free to try out Copilot or ChatGPT on their own initiative, most managers believed formal policies would ensure consistency with strategic goals, licensing compliance, and risk management. This blend of personal autonomy and the push for top-down oversight suggests that even within a single organization, social influence levels can differ. The understanding of these various levels of social influence can inform future research on the adoption of LLMs and other emerging technologies in software engineering, as well as organizational strategies for encouraging their appropriate use.

## 5.3 Augmentation vs. Automation.

Beyond examining short-term productivity gains and evolving workflows, recent studies highlight a broader shift in software engineering. Abrahão et al. [6] argue that the field is increasingly being redefined as a human-centric discipline, where ethical considerations and human judgment play an essential role alongside technical advancements. Similarly, Terragni et al. [24] discuss the emergence of a collaborative dynamic between human developers and generative AI, emphasizing how this interaction is not only accelerating routine tasks but also reshaping professional roles and development practices. These perspectives suggest that integrating GenAI in software engineering requires frameworks that balance

growing system autonomy with the necessity for human oversight, ensuring that innovation, trust, and accountability remain central to software development.

A key consideration in this transformation is whether GenAI should primarily augment human developers or seek to automate software engineering tasks entirely. Existing research suggests that current LLMs provide the greatest value when they function as assistive tools rather than as full replacements [23]. Our findings align with this, indicating that managers perceive LLMs as tools to enhance productivity, particularly in code generation and documentation, while emphasizing the need for human oversight for critical aspects like IP protection and security. This preference for augmentation suggests that organizational models that foster human-AI collaboration will be most effective. Such models might involve embedding AI specialists within development teams, establishing a clear process for validating AI outputs, and promoting a culture of continuous learning to adapt to evolving AI capabilities. This approach aligns with the concept of human-in-the-loop development, where GenAI-driven suggestions complement rather than replace domain expertise, improving efficiency without compromising quality or reliability.

While technical advancements may expand the scope of automation for well-structured and repetitive tasks, software engineering involves complex challenges, including system architecture, regulatory compliance, and ethical considerations. Fully autonomous GenAI-driven development would require robust safeguards and continuous managerial oversight to address these challenges effectively. Organizations must carefully balance efficiency gains with governance strategies and workforce adaptation [25]. This implies that roles will shift. Developers may focus more on validating, integrating, and refining AI-generated code. Testers might concentrate on complex edge cases and system-level validation. Managers will need to oversee AI governance and ensure ethical implementation. Future research should investigate how different organizational structures and evolving roles impact the long-term effectiveness and sustainability of GenAI adoption in software engineering. This means moving beyond immediate productivity effects to examine team dynamics, decision-making processes, and overall project success.

### 5.4 Threats to Validity

We acknowledge that several factors may influence the credibility and transferability of our findings:

- **Internal Validity:** Although multiple researchers facilitated and analyzed the workshop discussions, qualitative bias may still arise in interpreting managers' statements. Additionally, the presence of the researchers as facilitators may have influenced participants' openness and responses. To mitigate this, we used independent coding and cross-verification to consolidate themes.
- **External Validity:** Our sample of eight managers does not represent the full spectrum of organizations adopting LLMs. Still, including small, medium, and large entities provides a degree of diversity. Readers can assess the applicability of our insights to their contexts.
- **Construct Validity:** We leveraged established frameworks and literature on LLM-driven tasks to guide our analysis, but local practices and terminologies may have shaped participant responses.
- **Reliability:** Workshop outputs were not quantitatively measured and relied on voluntary self-reporting. Collaborative note-taking, audio recordings (where permissible), or a detailed audit trail might enhance reliability in future studies.
- **Temporal Validity:** The workshop was conducted in April 2024, capturing a snapshot of managerial perspectives. While this provides valuable insight into early LLM adoption, rapid technological advances may have shifted these views. We mitigated this limitation by triangulating our findings with recent literature, and we recommend follow-up studies to assess long-term trends.

## 6 Conclusions

### 6.1 Implications for Industry

Our study shows that while LLM-based tools are steadily integrated into code-centric tasks, many organizations remain cautious, particularly concerning non-trivial issues like intellectual property, security, and maintainability. For industry practitioners, developing guidelines on licensing, code ownership, and AI-driven development workflows is crucial. Collaborations between legal, security, and development teams can help mitigate perceived risks.

### 6.2 Implications for Research

Researchers need more extensive and continuous data on how LLMs shape the software lifecycle to assess AI's long-term impact. Beyond short-term productivity gains, future studies should examine code evolution, defect density, and organizational practices over time. By investigating how AI interventions integrate into real-world projects, the research community can more accurately map GenAI's broader effects on software engineering processes.

### 6.3 Conclusion and Future Work

This study offers empirical insights into how managers perceive the actual and potential uses of LLMs in software engineering. As GenAI tools mature, industries may witness an evolution in end-to-end workflows, ranging from requirements elicitation through maintenance and education. Yet the path to widespread adoption involves addressing tangible concerns around IP, reliability, and organizational readiness. Future research agendas should integrate both technical and organizational lenses, paving the way for robust AI-augmented software development life cycles. As organizations move beyond initial trials and internal prototypes to integrate LLMs into each stage of development, from concept to deployment, we anticipate that governance, collaborative workflows, and continuous education will become essential organizational practices. Moreover, following up with managers over time can provide a longitudinal view of how LLM adoption evolves and what challenges persist in real-world settings.

## References

[1] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. Yu, L. Sun, A survey of ai-generated content (aigc), ACM Comput. Surv. 57 (5) (Jan. 2025). doi:10.1145/3704262. URL https://doi.org/10.1145/3704262

[2] M. Al-kfairy, Strategic integration of generative ai in organizational settings: Applications, challenges and adoption requirements, IEEE Engineering Management Review (2025) 1–14doi:10.1109/EMR.2025.3534034.

[3] I. Ozkaya, Application of large language models to software engineering tasks: Opportunities, risks, and implications, IEEE Software 40 (3) (2023) 4–8. doi:10.1109/MS.2023.3248401.

[4] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, J. M. Zhang, Large language models for software engineering: Survey and open problems (2023) 31–53.

[5] D. Russo, Navigating the complexity of generative ai adoption in software engineering, ACM Trans. Softw. Eng. Methodol. 33 (5) (Jun. 2024). doi:10.1145/3652154.
URL https://doi.org/10.1145/3652154

[6] S. Abrahão, J. Grundy, M. Pezzè, M.-A. Storey, D. Andrew Tamburri, Software engineering by and for humans in an ai era, ACM Trans. Softw. Eng. Methodol.Just Accepted (Feb. 2025). doi:10.1145/3715111.
URL https://doi.org/10.1145/3715111

[7] Z. Zheng, K. Ning, Q. Zhong, J. Chen, W. Chen, L. Guo, W. Wang, Y. Wang, Towards an understanding of large language models in software engineering tasks, Empirical Software Engineering 30 (2) (2025) 50. doi:10.1007/s10664-024-10602-0.

[8] R. Ulfsnes, N. B. Moe, V. Stray, M. Skarpen, Transforming software development with generative ai: Empirical insights on collaboration and workflow, in: A. Nguyen-Duc, P. Abrahamsson, F. Khomh (Eds.), Generative AI for Effective Software Development, Springer Nature Switzerland, Cham, 2024, pp. 219–234. doi:10.1007/978-3-031-55642-5_10.
URL https://doi.org/10.1007/978-3-031-55642-5_10

[9] K. Prasad Agrawal, Towards adoption of generative ai in organizational settings, Journal of Computer Information Systems 64 (5) (2024) 636–651. doi:10.1080/08874417.2023.2240744.

[10] D. Russo, S. Baltes, N. van Berkel, P. Avgeriou, F. Calefato, B. Cabrero-Daniel, G. Catolino, J. Cito, N. Ernst, T. Fritz, H. Hata, R. Holmes, M. Izadi, F. Khomh, M. B. Kjærgaard, G. Liebel, A. L. Lafuente, S. Lambiase, W. Maalej, G. Murphy, N. B. Moe, G. O'Brien, E. Paja, M. Pezzè, J. S. Persson, R. Prikladnicki, P. Ralph, M. Robillard, T. R. Silva, K.-J. Stol, M.-A. Storey, V. Stray, P. Tell, C. Treude, B. Vasilescu, Generative ai in software engineering must be human-centered: The copenhagen manifesto, J. Syst. Softw. 216 (C) (Oct. 2024). doi:10.1016/j.jss.2024.112115.
URL https://doi.org/10.1016/j.jss.2024.112115

[11] A. Carleton, D. Falessi, H. Zhang, X. Xia, Generative ai: Redefining the future of software engineering, IEEE Software 41 (6) (2024) 34–37. doi:10.1109/MS.2024.3441889.

[12] J. Jahić, A. Sami, State of practice: Llms in software engineering and software architecture, in: 2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C), 2024, pp. 311–318. doi:10.1109/ICSA-C63560.2024.00059.

[13] S. Lambiase, G. Catolino, F. Palomba, F. Ferrucci, D. Russo, Investigating the role of cultural values in adopting large language models for software engineering (2024). arXiv:2409.05055.
URL https://arxiv.org/abs/2409.05055

[14] S. Betz, B. Penzenstadler, With great power comes great responsibility: The role of software engineers, ACM Trans. Softw. Eng. Methodol.Just Accepted (Jan. 2025). doi:10.1145/3715112.
URL https://doi.org/10.1145/3715112

[15] Large language models for software engineering: A systematic literature review, ACM Trans. Softw. Eng. Methodol. 33 (8) (Dec. 2024). doi:10.1145/3695988.
URL https://doi.org/10.1145/3695988

[16] M. Coutinho, L. Marques, A. Santos, M. Dahia, C. França, R. de Souza Santos, The role of generative ai in software development productivity: A pilot case study (2024). doi:10.1145/3664646.3664773.
URL https://doi.org/10.1145/3664646.3664773

[17] R. Tufano, A. Mastropaolo, F. Pepe, O. Dabic, M. Di Penta, G. Bavota, Unveiling chatgpt's usage in open source projects: A mining-based study, in: Proceedings of the 21st International Conference on Mining Software Repositories, MSR '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 571–583. doi:10.1145/3643991.3644918.
URL https://doi.org/10.1145/3643991.3644918

[18] R. Khojah, M. Mohamad, P. Leitner, F. G. de Oliveira Neto, Beyond code generation: An observational study of chatgpt usage in software engineering practice, Vol. 1, Association for Computing Machinery, New York, NY, USA, 2024. doi:10.1145/3660788.
URL https://doi.org/10.1145/3660788

[19] J. Sallou, T. Durieux, A. Panichella, Breaking the silence: the threats of using llms in software engineering, in: Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER'24, Association for Computing Machinery, New York, NY, USA, 2024, p. 102–106. doi:10.1145/3639476.3639764.
URL https://doi.org/10.1145/3639476.3639764

[20] K. Thoring, R. Mueller, P. Badke-Schaub, Workshops as a Research Method: Guidelines for Designing and Evaluating Artifacts Through Workshops, in: Hawaii International Conference on System Sciences, 2020. doi:10.24251/HICSS.2020.620.

[21] A. Moharil, A. Sharma, Tabasco: A transformer based contextualization toolkit, Sci. Comput. Program. 230 (C) (Aug. 2023). doi:10.1016/j.scico.2023.102994.
URL https://doi.org/10.1016/j.scico.2023.102994

[22] B. Baudry, K. Etemadi, S. Fang, Y. Gamage, Y. Liu, Y. Liu, M. Monperrus, J. Ron, A. Silva, D. Tiwari, Generative ai to generate test data generators, IEEE Software 41 (6) (2024) 55–64. doi:10.1109/MS.2024.3418570.

[23] S. Barke, M. B. James, N. Polikarpova, Grounded copilot: How programmers interact with code-generating models, Proc. ACM Program. Lang. 7 (OOPSLA1) (Apr. 2023). doi:10.1145/3586030.
URL https://doi.org/10.1145/3586030

[24] V. Terragni, A. Vella, P. Roop, K. Blincoe, The future of ai-driven software engineering, ACM Trans. Softw. Eng. Methodol.Just Accepted (Jan. 2025). doi:10.1145/3715003.
URL https://doi.org/10.1145/3715003

[25] J. Jöhnk, M. Weißert, K. Wyrtki, Ready or not, ai comes—an interview study of organizational ai readiness factors, Business & Information Systems Engineering 63 (1) (2021) 5–20. doi:10.1007/s12599-020-00676-7.