Forecasting Developer Environments with GenAI: A Research Perspective

Raula Gaikovina Kula[1], Christoph Treude[2], Xing Hu[3], Sebastian Baltes[4], Earl T. Barr[5], Kelly Blincoe[6], Fabio Calefato[7], Junjie Chen[8], Marc Cheong[9], Youmei Fan[10], Daniel M. German[11], Marco Gerosa[12], Jin Guo[13], Shinpei Hayashi[14], Robert Hirschfeld[15,16], Reid Holmes[17], Yintong Huo[2], Takashi Kobayashi[14], Michele Lanza[18], Zhongxin Liu[3], Olivier Nourry[1], Nicole Novielli[7], Denys Poshyvanyk[20], Shinobu Saito[24], Kazumasa Shimari[10], Igor Steinmacher[12], Mairieli Wessel[21], Markus Wagner[22], Annie Vella[6], Laurie Williams[23], Xin Xia[3]

[1]The University of Osaka, [2]Singapore Management University, [3]Zhejiang University, [4]Heidelberg University, [5]University College London, [6]University of Auckland, [7]University of Bari, [8]Tianjin University, [9]University of Melbourne, [10]Nara Institute of Science and Technology, [11]University of Victoria, [12]Northern Arizona University, [13]McGill University, [14]Institute of Science Tokyo, [15]Hasso Plattner Institut, [16]University of Potsdam, [17]University of British Columbia, [18]Università della Svizzera italiana, [19]University of Bari, [20]William and Mary, [21]Radboud University, [22]Monash University, [23]North Carolina State University, [24]NTT Computer and Data Science Laboratories

Abstract

Generative Artificial Intelligence (GenAI) models are achieving remarkable performance in various tasks, including code generation, testing, code review, and program repair. The ability to increase the level of abstraction away from writing code has the potential to change the Human-AI interaction within the integrated development environment (IDE). To explore the impact of GenAI on IDEs, 33 experts from the Software Engineering, Artificial Intelligence, and Human-Computer Interaction domains gathered to discuss challenges and opportunities at Shonan Meeting 222, a four-day intensive research meeting. Four themes emerged as areas of interest to the researchers: to what extent the IDE will evolve to solve current problems, how the IDE can lead the charge for a new paradigm, how the human role will change, and radical ideas of how the IDE will look in the future.

CCS Concepts

• **Software and its engineering → Development frameworks and environments**.

Keywords

Foundation Models, Development frameworks, Human-AI Collaboration, Generative Artificial Intelligence

ACM Reference Format:

Raula Gaikovina Kula, Christoph Treude, Xing Hu, Sebastian Baltes, Earl T. Barr, Kelly Blincoe, Fabio Calefato, Junjie Chen, Marc Cheong, Youmei Fan, Daniel M. German, Marco Gerosa, Jin Guo, Shinpei Hayashi, Robert Hirschfeld, Reid Holmes, Yintong Huo, Takashi Kobayashi, Michele Lanza, Zhongxin Liu, Olivier Nourry, Nicole Novielli, Denys Poshyvanyk, Shinobu Saito, Kazumasa Shimari, Igor Steinmacher, Mairieli Wessel, Markus Wagner, Annie Vella, Laurie Williams, Xin Xia. 2026. Forecasting Developer Environments with GenAI: A Research Perspective. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Generative Artificial Intelligence (GenAI) models are already achieving remarkable performance in various software engineering tasks such as code generation, testing, code review, and program repair [1–5]. Furthermore, the rapid adoption of GenAI has compelled both practitioners and researchers to integrate these tools, often without fully understanding their long-term consequences [6, 7].

To understand the impact of GenAI on the Integrated Development Environment (IDE), 33 experts from the three domains of Software Engineering (SE), Artificial Intelligence (AI), and Human-Computer Interaction (HCI) gathered to discuss the grand challenges and opportunities at Shonan Meeting 222 "*The Future of Development Environments with AI Foundation Models*"[1]. To help organize the Shonan, we sent out a pre-event survey that highlighted five questions that researchers felt that were important when discussing the future of development environments:

- Which software development tasks should GenAI handle?
- How should GenAI be integrated into IDE features?
- What role remains for humans in software development?
- Do we still need development environments?
- Your boldest claim for software engineering in 2050?

Based on these five questions above, participants then developed and presented their own impressions of what the future of IDEs would look like. As the presentations progressed, it was apparent that four themes emerged from these intensive discussions, which we discuss in the subsequent sections.

[1]https://shonan.nii.ac.jp/seminars/222/

2 A Changing IDE, Solving the Same Problems

Some researchers view AI-augmented IDEs as an evolutionary step rather than a radical overhaul of software engineering practice [8, 9]. The core tasks—design, coding, debugging, testing—remain unchanged; GenAI simply automates the repetitive or mechanical aspects, freeing developers to devote more time to high-level design and creative problem-solving. This automation is expected to streamline workflows, reduce boilerplate work, and accelerate delivery cycles, thereby enhancing overall productivity [10, 11].

Despite these benefits, significant hurdles persist. Aligning training data across the software engineering and AI domains demands careful curation, while divergent terminologies—such as symbolic versus neural representations [12]—require interdisciplinary dialogue to create shared ontologies. Moreover, GenAI is increasingly taking care of low-level technicalities and idiosyncrasies, pushing abstraction layers higher within IDEs. This shift promises greater developer autonomy but also risks obscuring underlying implementation details if not managed thoughtfully [13–16].

Ultimately, programming must remain an immersive, collaborative human activity. As GenAI handles more routine tasks, developers will be left to intervene only during rare yet critical moments—debugging complex issues or architecting scalable systems [17–19], as recent research on developer-AI collaboration shows how interaction patterns are shifting [20, 21]. Ensuring that automation enhances rather than diminishes creativity requires IDE designs that encourage intentional interaction and maintain the developer's central role in decision-making [22].

Lesson Learnt. The key lesson is that although the technologies are changing, the basic components of software development and its key guiding principles remain the same. We are simply moving to a higher level of abstraction, and all tedious work will be handled by the computing power, leaving tasks of critical thinking.

3 A New Paradigm That Begins from the IDE

The integration of Generative AI into IDEs is expected to spawn a host of novel research avenues, spanning causal inference, legal implications, and software engineering education [23]. By treating code as an artifact that can be questioned, challenged, and re-generated, these systems will enable developers to explore alternative implementations faster, fostering a turn-taking dynamic between human intent and AI suggestion [24, 25]. This raises the need for specialized tools that support GenAI-driven code: knowledge and context management, orchestrated agent swarms executing complex workflows, automated checkpointing and repair mechanisms, and management of "LLM technical debt" that arises when large language models become tightly coupled to production systems [26, 27].

Moreover, as language models begin to generate increasingly sophisticated and self-correcting code, developers may find themselves acting more like digital gardeners than traditional programmers [28, 29]. Systems could evolve autonomously, with new tools emerging to address the unique problems posed by such self-evolving software—ranging from continuous verification of causality to ensuring compliance with evolving legal standards [30, 31]. In short, GenAI will not only augment existing IDE capabilities but also necessitate entirely new toolchains designed for a future where code is both created and maintained autonomously by intelligent agents with access to the right knowledge, ultimately realising a vision in which the IDE becomes genuinely integrated rather than the largely isolated environment it has often felt like so far [32, 33].

Lessons Learnt. Rather than just being an evolution, the IDE will cause us to rethink how we build software. Ideas like evolving software might have to be scraped to on-demand software that can be easily thrown away. The IDE will cater for a broader demographic of users, and software could be self-evolving without humans constantly intervening. On the other hand, this changing IDE might also bring new problems that were not faced by software engineering developers, opening up new avenues for research.

4 The Human Role Reimagined

The contemporary developer increasingly assumes a managerial role in the software development ecosystem [34]. Rather than merely implementing code, developers must now grapple with *comprehension debt*—the cognitive cost of understanding legacy systems, unfamiliar frameworks, and evolving requirements [35]. A personalized command center that aggregates relevant metrics, documentation, and contextual insights can mitigate this burden, allowing the developer to act as a project manager who orchestrates rather than writes code [32, 36].

In this paradigm shift, the human actor functions more as an *author of intent*, a guide for the system's autonomous components, and a facilitator of collaboration among distributed agents [37–39]. The IDE should therefore support clarifying ambiguous specifications, enforcing quality assurance, and validating design decisions without requiring the developer to engage in low-level implementation details. By abstracting routine tasks, the environment frees developers to focus on higher-order problem-solving [32, 40].

Design principles for such an IDE must prioritize assistance over frustration. Cognitive studies suggest that excessive mental load can impair creativity and lead to burnout [41, 42]; consequently, interfaces should be intuitive, provide adaptive scaffolding, and offer clear pathways for intellectual exploration. Transparency and trust are essential: developers need to understand how the system arrives at its recommendations or predictions, which in turn fosters confidence in automated decisions [43, 44].

Ethical considerations also surface when humans delegate increasingly complex tasks to AI agents [45, 46]. Biases embedded in training data can propagate through decision-making pipelines, potentially harming users or reinforcing unfair practices [47–49]. The IDE should expose these biases and provide mechanisms for human oversight [50]. Simultaneously, it must support continuous skills development, enabling developers to evolve from mere coders into system architects who design the overall structure rather than write every line of code [36, 51, 52].

Finally, a mutual *theory of mind* between developer and AI is crucial: each party should model the other's intentions, constraints, and preferences [53, 54]. This reciprocal understanding allows for more natural collaboration and reduces miscommunication. As the profession matures, the role of software engineers—sometimes dubbed "source argonauts"—will shift: while their individual share of code may shrink, their influence over architectural decisions and

strategic direction will grow, reflecting a new hierarchy within the development ecosystem [55, 56].

Lessons Learnt. How developers interact with the machine to build software will change. The relationship may evolve from simply giving instructions to now managing what the machine recommends. This changing relationship may alter cognitive load and decision-making. Thus the IDE will need to be robust to accommodate clear communication and verification of intentions, constraints, and preferences. Since building software is a collaborative activity, the IDE will also have to facilitate interactions between all members of the team, with some of them maybe not human.

5 Futurecasting the 2050 IDE

This *futuring* exercise imagines the context of programming in the future: what can be hypothetically *seen* in 2050, without being overly concerned with "how to get there". We jump to the year 2050 to dig into components to understand what we see.

Immersion and User experience. Walking into the room, the system powers on and immediately fills the space. It clearly distinguishes itself from its surroundings yet remains immersive enough for a user or creator to recognize familiar functions; this suggests that developers of that era had reached some consensus about how invasive the environment should be to meet users' needs. By waving their hands around, an entire simulation materializes, and intuitive controls allow navigation through the complex system. A time-travel feature lets a user simulate changes to understand their effects, while real-time visualization of collaborators shows how all these modifications are being introduced into the system.

Components of the IDE. Digging into the artifacts of 2050, we find that they are equipped with multiple AI agents and multimodal interfaces. Among these is a super-AI that serves as the core CPU, orchestrating the execution of the artifact. By integrating these AIs, the environment can evolve and repair bugs autonomously without explicit lines of code from humans. The compiler is designed to detect ambiguities in natural language and to be syntax-agnostic for AI agents, ensuring that they run without errors. The distinction between creating and using software no longer exists. People generate and modify software components directly through natural language, gestures, or biological and contextual signals. Every interaction with the system can alter its behavior, so users effectively become developers. The development environment interprets intent across multiple forms of input, allowing software to adapt instantly to new needs. Programming becomes a shared process between human and system, unfolding continuously through everyday use rather than within discrete development phases.

Taking inspiration from sci-fi franchises such as *Star Trek* and *Warhammer 40k*, the holodeck metaphor applies to the IDE of the future: "wireless communion with the machine spirit". What-if scenarios—e.g., "What if Google Maps routed everyone down the same road?"—are answered using full-scale digital twins. In a boon for agility and rapid prototyping, dials for *fidelity* and *scope* allow rapid re-evaluation of the current scenario. Edge cases, exceptions, and unintended changes over time are best represented as clusters of unique outcomes; what is "unintended" will be revealed upfront if the system records everything.

Feedback Loops. Motivated by advances in quantum computing, many states of the system can be explored in parallel in a world where compute power is no longer a constraint, enabling abstractions and jumps when needed. Multiple users (or developers) can engage with the system together; tools exist for one participant to "enter" the system's internal state and make edits, thereby blurring the boundaries between creator and user. From observing unintended outcomes in the simulator to steering the program toward more intended behaviors, an effective feedback loop is essential. As the system becomes increasingly complex, this loop must become diagnostic, helping developers reason about the immediate next step. Because the line between using and programming software blurs, the level of expertise required diversifies; consequently, the granularity and focus of the feedback are automatically adjusted to match each developer's technical background. Importantly, the impact observed from the simulator is both social and technical, each situated on different but interconnected dimensions. The feedback loop is multidimensional: it encourages debate and reflection on complex social and value-laden issues, explicitly links social and technical concerns, and translates them into concrete guidelines for implementation. Finally, it provides mechanisms for coordination and norm establishment to resolve conflicts and debates raised by the diverse parties involved in technological advancement.

Lessons of the exercise. The key lesson of the exercise is that it might be time to let go of conventional notions of files and folder organization, version control, and source code as the primary artifact. A future beyond the keyboard as the main mode of input is possible, while collaboration between humans remains foundational to IDE.

6 In Closing

We forecast developer environments to undergo major changes, however, the themes reveal that our understanding of the direction of change is still far from set in stone. While prevailing trends suggest possible directions for their role, design and functionality, researchers must continually assess whether these trajectories advance scientific inquiry. This critical appraisal ensures that innovations align with the needs of both industry and academia rather than merely following market hype. Envisioning the IDE of 2050 invites us to rethink what it means to '*develop*' software, with human creativity and judgment remaining central.

References

[1] Y. Deng, C. S. Xia, C. Yang, S. D. Zhang, S. Yang, and L. Zhang, "Large language models are edge-case generators: Crafting unusual programs for fuzzing deep learning libraries," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[2] C. S. Xia, Y. Wei, and L. Zhang, "Automated program repair in the era of large pre-trained language models," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 1482–1494, 2023.

[3] Z. Yuan, M. Liu, S. Ding, K. Wang, Y. Chen, X. Peng, and Y. Lou, "Evaluating and improving chatgpt for unit test generation," *Proc. ACM Softw. Eng.*, vol. 1, July 2024.

[4] C. Gao, X. Hu, S. Gao, X. Xia, and Z. Jin, "The current challenges of software engineering in the era of large language models," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 5, pp. 1–30, 2025.

[5] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? assessing the security of github copilot's code contributions," *Communications of the ACM*, vol. 68, no. 2, pp. 96–105, 2025.

[6] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in

Chi conference on human factors in computing systems extended abstracts, pp. 1–7, 2022.
[7] S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA1, pp. 85–111, 2023.
[8] I. Ozkaya, "Application of large language models to software engineering tasks: Opportunities, risks, and implications," *IEEE Software*, vol. 40, no. 3, pp. 4–8, 2023.
[9] I. Ozkaya, "The next frontier in software development: Ai-augmented software development processes," *IEEE Software*, vol. 40, no. 4, pp. 4–9, 2023.
[10] M. Coutinho, L. Marques, A. Santos, M. Dahia, C. França, and R. de Souza Santos, "The role of generative ai in software development productivity: A pilot case study," in *Proceedings of the 1st ACM International Conference on AI-Powered Software*, pp. 131–138, 2024.
[11] L. Yu, "Paradigm shift on coding productivity using genai," *arXiv preprint arXiv:2504.18404*, 2025.
[12] X. Zhang and V. S. Sheng, "Bridging the gap: representation spaces in neuro-symbolic ai," *arXiv preprint arXiv:2411.04393*, 2024.
[13] A. Sergeyuk, I. Zakharov, E. Koshchenko, and M. Izadi, "Human-ai experience in integrated development environments: A systematic literature review," *arXiv preprint arXiv:2503.06195*, 2025.
[14] V. Chen, A. Talwalkar, R. Brennan, and G. Neubig, "Code with me or for me? how increasing ai automation transforms developer workflows," *arXiv preprint arXiv:2507.08149*, 2025.
[15] R. Ulfsnes, N. B. Moe, V. Stray, and M. Skarpen, "Transforming software development with generative ai: Empirical insights on collaboration and workflow," in *Generative AI for effective software development*, pp. 219–234, Springer, 2024.
[16] C. Treude and R. G. Kula, "Interacting with ai reasoning models: Harnessing "thoughts" for ai-driven software engineering," 2025.
[17] B. Houck, T. Lowdermilk, C. Beyer, S. Clarke, and B. Hanrahan, "The space of ai: Real-world lessons on ai's impact on developers," *arXiv preprint arXiv:2508.00178*, 2025.
[18] A. Kumar, Y. Bajpai, S. Gulwani, G. Soares, and E. Murphy-Hill, "How developers use ai agents: When they work, when they don't, and why," *arXiv preprint arXiv:2506.12347*, 2025.
[19] S. Pan, L. Wang, T. Zhang, Z. Xing, Y. Zhao, Q. Lu, and X. Sun, ""i don't use ai for everything": Exploring utility, attitude, and responsibility of ai-empowered tools in software development," *arXiv preprint arXiv:2409.13343*, 2024.
[20] C. Treude and M. A. Gerosa, "How developers interact with ai: A taxonomy of human-ai collaboration in software engineering," in *2025 IEEE/ACM Second International Conference on AI Foundation Models and Software Engineering (Forge)*, pp. 236–240, IEEE, 2025.
[21] T. Settewong, Y. Fan, R. G. Kula, and K. Matsumoto, "Human to document, ai to code: Three case studies of comparing genai for notebook competitions," 2025.
[22] S. Inman, A. Murillo, S. D'Angelo, A. Brown, and C. Green, "Seamful AI for creative software engineering," *IEEE Software*, 2025. Conference Name: IEEE Software.
[23] A. Nguyen-Duc, B. Cabrero-Daniel, A. Przybylek, C. Arora, D. Khanna, T. Herda, U. Rafiq, J. Melegati, E. Guerra, K.-K. Kemell, *et al.*, "Generative artificial intelligence for software engineering—a research agenda," *Software: Practice and Experience*, 2025.
[24] K. Ferdowsi, R. Huang, M. B. James, N. Polikarpova, and S. Lerner, "Validating ai-generated code with live programming," in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pp. 1–8, 2024.
[25] R. Rojpaisarnkit, Y. Fan, K. Matsumoto, and R. G. Kula, "How natural language proficiency shapes genai code for software engineering tasks," *IEEE Software*, pp. 1–7, 2025.
[26] A. Menshawy, Z. Nawaz, and M. Fahmy, "Navigating challenges and technical debt in large language models deployment," in *Proceedings of the 4th Workshop on Machine Learning and Systems*, pp. 192–199, 2024.
[27] A. Aljohani and H. Do, "Promptdebt: A comprehensive study of technical debt across llm projects," *arXiv preprint arXiv:2509.20497*, 2025.
[28] K. Qiu, N. Puccinelli, M. Ciniselli, and L. Di Grazia, "From today's code to tomorrow's symphony: The ai transformation of developer's routine by 2030," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 5, pp. 1–17, 2025.
[29] R. G. Kula and C. Treude, "The shift from writing to pruning software: A bonsai-inspired ide for reshaping ai generated code," 2025.
[30] D. Weyns, T. Bäck, R. Vidal, X. Yao, and A. N. Belbachir, "The vision of self-evolving computing systems," *Journal of Integrated Design and Process Science*, vol. 26, no. 3-4, pp. 351–367, 2023.
[31] L. Cai, Y. Ren, Y. Zhang, and J. Li, "Ai-driven self-evolving software: A promising path toward software automation," *arXiv preprint arXiv:2510.00591*, 2025.
[32] M. Marron, "A new generation of intelligent development environments," in *Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments*, pp. 43–46, 2024.
[33] M. Tufano, A. Agarwal, J. Jang, R. Z. Moghaddam, and N. Sundaresan, "Autodev: Automated ai-driven development," *arXiv preprint arXiv:2403.08299*, 2024.
[34] E. Kalliamvakou, C. Bird, T. Zimmermann, A. Begel, R. DeLine, and D. M. German, "What makes a great manager of software engineers?," *IEEE Transactions on Software Engineering*, vol. 45, no. 1, pp. 87–106, 2017.
[35] E. Agrawal, O. Alam, C. Goenka, M. Iyer, I. Moise, A. Pandian, and B. Paul, "Code compass: A study on the challenges of navigating unfamiliar codebases," *arXiv preprint arXiv:2405.06271*, 2024.
[36] A. E. Hassan, G. A. Oliva, D. Lin, B. Chen, Z. Ming, *et al.*, "Towards ai-native software engineering (se 3.0): A vision and a challenge roadmap," *arXiv preprint arXiv:2410.06107*, 2024.
[37] C. Treude and M.-A. Storey, "Generative ai and empirical software engineering: A paradigm shift," *arXiv preprint arXiv:2502.08108*, 2025.
[38] Z. Rasheed, M. Waseem, M. A. Sami, K.-K. Kemell, A. Ahmad, A. N. Duc, K. Systä, and P. Abrahamsson, "Autonomous agents in software development: A vision paper," in *International Conference on Agile Software Development*, pp. 15–23, Springer Nature Switzerland Cham, 2024.
[39] W. Takerngsaksiri, J. Pasuksmit, P. Thongtanunam, C. Tantithamthavorn, R. Zhang, F. Jiang, J. Li, E. Cook, K. Chen, and M. Wu, "Human-in-the-loop software development agents," in *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 342–352, IEEE, 2025.
[40] A. E. Hassan, H. Li, D. Lin, B. Adams, T.-H. Chen, Y. Kashiwa, and D. Qiu, "Agentic software engineering: Foundational pillars and a research roadmap," *arXiv preprint arXiv:2509.06216*, 2025.
[41] L. Gonçales, K. Farias, B. da Silva, and J. Fessler, "Measuring the cognitive load of software developers: A systematic mapping study," in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, pp. 42–52, IEEE, 2019.
[42] S. Chandrasekaran, "Enhancing developer experience by reducing cognitive load: A focus on minimization strategies," *International Journal of Computer Trends and Technology*, vol. 72, no. 1, pp. 99–103, 2024.
[43] R. Wang, R. Cheng, D. Ford, and T. Zimmermann, "Investigating and designing for trust in ai-powered code generation tools," in *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1475–1493, 2024.
[44] S. Baltes, T. Speith, B. Chiteri, S. Mohsenimofidi, S. Chakraborty, and D. Buschek, "Rethinking trust in ai assistants for software development: A critical review," *arXiv preprint arXiv:2504.12461*, 2025.
[45] G. Dodig-Crnkovic, G. Basti, and T. Holstein, "Delegating responsibilities to intelligent autonomous systems: Challenges and benefits," *Journal of Bioethical Inquiry*, pp. 1–8, 2025.
[46] S. Abrahão, J. Grundy, M. Pezzè, M.-A. Storey, and D. A. Tamburri, "Software engineering by and for humans in an ai era," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 5, pp. 1–46, 2025.
[47] Y. Brun and A. Meliou, "Software fairness," in *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pp. 754–759, 2018.
[48] C. Treude and H. Hata, "She elicits requirements and he tests: Software engineering gender bias in large language models," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, pp. 624–629, IEEE, 2023.
[49] M. Sami, A. Sami, and P. Barclay, "A case study of fairness in generated images of large language models for software engineering tasks," in *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 391–396, IEEE, 2023.
[50] M. Atemkeng, S. Hamlomo, B. Welman, N. Oyetunji, P. Ataei, and J. L. K. Fendji, "Ethics of software programming with generative ai: is programming without generative ai always radical?," *arXiv preprint arXiv:2408.10554*, 2024.
[51] A. Birillo, M. Tigina, Z. Kurbatova, A. Potriasaeva, I. Vlasov, V. Ovchinnikov, and I. Gerasimov, "Bridging education and development: Ides as interactive learning platforms," in *Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments*, pp. 53–58, 2024.
[52] A. Sergeyuk, E. Koshchenko, I. Zakharov, T. Bryksin, and M. Izadi, "The design space of in-ide human-ai experience," *arXiv preprint arXiv:2410.08676*, 2024.
[53] S. Zhang, X. Wang, W. Zhang, Y. Chen, L. Gao, D. Wang, W. Zhang, X. Wang, and Y. Wen, "Mutual theory of mind in human-ai collaboration: An empirical study with llm-driven ai agents in a real-time shared workspace task," *arXiv preprint arXiv:2409.08811*, 2024.
[54] Q. Wang, K. Saha, E. Gregori, D. Joyner, and A. Goel, "Towards mutual theory of mind in human-ai interaction: How language reflects what students perceive about a virtual teaching assistant," in *Proceedings of the 2021 CHI conference on human factors in computing systems*, pp. 1–14, 2021.
[55] S. Betz and B. Penzenstadler, "With great power comes great responsibility: The role of software engineers," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 5, pp. 1–21, 2025.
[56] E. Meade, E. O'Keeffe, N. Lyons, D. Lynch, M. Yilmaz, U. Gulec, R. V. O'Connor, and P. M. Clarke, "The changing role of the software engineer," in *European conference on software process improvement*, pp. 682–694, Springer, 2019.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009