Forecasting Developer Environments with GenAI: A Research Perspective

Raula Gaikovina Kula
The University of Osaka
Osaka, Japan
raula-k@ist.osaka-u.ac.jp

Christoph Treude
Singapore Management University
Singapore, Singapore
ctreude@smu.edu.sg

Xing Hu
Zhejiang University
Hangzhou, China
xinghu@zju.edu.cn

Sebastian Baltes
Heidelberg University
Heidelberg, Germany
sebastian.baltes@uni-heidelberg.de

Earl T. Barr
University College London
London, UK
e.barr@ucl.ac.uk

Kelly Blincoe
University of Auckland
Auckland, New Zealand
k.blincoe@auckland.ac.nz

Fabio Calefato
University of Bari
Bari, Italy
fabio.calefato@uniba.it

Junjie Chen
Tianjin University
Tianjin, China
junjiechen@tju.edu.cn

Marc Cheong
University of Melbourne
Melbourne, Australia
marc.cheong@unimelb.edu.au

Youmei Fan
Nara Institute of Science and Technology
Nara, Japan
fan.youmei.fs2@is.naist.jp

Daniel M. German
University of Victoria
Victoria, BC, Canada
dmg@uvic.ca

Marco Gerosa
Northern Arizona University
Flagstaff, AZ, USA
Marco.Gerosa@nau.edu

Jin L.C. Guo
McGill University
Montreal, QC, Canada
jguo@cs.mcgill.ca

Shinpei Hayashi
Institute of Science Tokyo
Tokyo, Japan
hayashi@comp.isct.ac.jp

Robert Hirschfeld
Hasso Plattner Institute & University of Potsdam
Potsdam, Germany
robert.hirschfeld@hpi.uni-potsdam.de

Reid Holmes
University of British Columbia
Vancouver, BC, Canada
rtholmes@cs.ubc.ca

Yintong Huo
Singapore Management University
Singapore, Singapore
ythuo@smu.edu.sg

Takashi Kobayashi
Institute of Science Tokyo
Tokyo, Japan
tkobaya@comp.isct.ac.jp

Michele Lanza
Software Institute, USI, Lugano
Lugano, Switzerland
michele.lanza@usi.ch

Zhongxin Liu
Zhejiang University
Hangzhou, China
liu_zx@zju.edu.cn

Olivier Nourry
The University of Osaka
Osaka, Japan
nourry@ist.osaka-u.ac.jp

Nicole Novielli
University of Bari
Bari, Italy
nicole.novielli@uniba.it

Denys Poshyvanyk
William & Mary
Williamsburg, VA, USA
dposhyvanyk@gmail.com

Shinobu Saito
NTT Computer and Data Science Laboratories
Tokyo, Japan
shinobu.saito@ntt.com

Kazumasa Shimari
Nara Institute of Science and Technology
Nara, Japan
k.shimari@is.naist.jp

Igor Steinmacher
Northern Arizona University
Flagstaff, AZ, USA
igor.steinmacher@nau.edu

Mairieli Wessel
Radboud University
Nijmegen, Netherlands
mairieli.wessel@ru.nl

Markus Wagner
Monash University
Melbourne, Australia
markus.wagner@monash.edu

Annie Vella
University of Auckland
Auckland, New Zealand
annie.luxton@gmail.com

Laurie Williams
North Carolina State University
Raleigh, NC, USA
lawilli3@ncsu.edu

Xin Xia
Zhejiang University
Hangzhou, China
xin.xia@zju.edu.cn

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 for researchers and practitioners.

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 L.C. 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, and Xin Xia. 2026. Forecasting Developer Environments with GenAI: A Research Perspective. In *IDE '26: 3rd International Workshop on Integrated Development Environments, April 18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

IDE '26, Rio de Janeiro, Brazil

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 [14, 17, 33, 52, 54]. 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, 48].

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 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, 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. It is important to note that participants did not all agree with the description of the themes, with several disagreements and divergent worldviews between the themes. For example, the participants were split between themes one and two.

2 Theme One: 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 [30,

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

31]. 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 [13, 53].

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 [56]—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 [12, 39, 44, 47].

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 [21, 25, 32], as recent research on developer-AI collaboration shows how interaction patterns are shifting [40, 42]. 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].

Lessons 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 Theme Two: 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 [29]. 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 [16, 36]. 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 [3, 28].

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 [24, 34]. 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 [10, 51]. 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 [26, 46].

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 replaced by 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 previously faced by software developers, opening up new avenues for research.

4 Theme Three: The Human Role Reimagined

The contemporary developer increasingly assumes a managerial role in the software development ecosystem [23]. Rather than merely implementing code, developers must now grapple with *comprehension debt*—the cognitive cost of understanding legacy systems, unfamiliar frameworks, and evolving requirements [2]. 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 [20, 26].

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 [35, 41, 45]. 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 [19, 26].

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 [11, 18]; 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 [5, 50].

Ethical considerations also surface when humans delegate increasingly complex tasks to AI agents [1, 15]. Biases embedded in training data can propagate through decision-making pipelines, potentially harming users or reinforcing unfair practices [9, 37, 43]. The IDE should expose these biases and provide mechanisms for human oversight [4]. 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 [8, 20, 38].

Finally, a mutual *theory of mind* between developer and AI is crucial: each party should model the other's intentions, constraints, and preferences [49, 55]. 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 [7, 27].

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 potentially not human.

5 Theme Four: 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".

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 in the IDE. 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.

Lessons Learnt. 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 the 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.

There are other aspects of the IDE that were not discussed during the Shonan, but also deserve attention. For example, the legal aspects of using AI-driven IDEs generating all kinds of artifacts brings forth questions such as the responsibility if an AI-driven IDE creates erroneous code (or wrong designs or insufficient tests) that leads to severe consequences (e.g., loss of money, harm to humans). Still, we hope that these themes invite us to rethink what it means to '*develop*' software, with human creativity and judgment remaining central.

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 Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–46.

[2] Ekansh Agrawal, Omair Alam, Chetan Goenka, Medha Iyer, Isabela Moise, Ashish Pandian, and Bren Paul. 2024. Code compass: A study on the challenges of navigating unfamiliar codebases. *arXiv preprint arXiv:2405.06271* (2024).

[3] Ahmed Aljohani and Hyunsook Do. 2025. PromptDebt: A Comprehensive Study of Technical Debt Across LLM Projects. *arXiv preprint arXiv:2509.20497* (2025).

[4] Marcellin Atemkeng, Sisipho Hamlomo, Brian Welman, Nicole Oyetunji, Pouya Ataei, and Jean Louis KE Fendji. 2024. Ethics of software programming with generative AI: is programming without generative AI always radical? *arXiv preprint arXiv:2408.10554* (2024).

[5] Sebastian Baltes, Timo Speith, Brenda Chiteri, Seyedmoein Mohsenimofidi, Shalini Chakraborty, and Daniel Buschek. 2025. Rethinking Trust in AI Assistants for Software Development: A Critical Review. *arXiv preprint arXiv:2504.12461* (2025).

[6] Shraddha Barke, Michael B James, and Nadia Polikarpova. 2023. Grounded copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages* 7, OOPSLA1 (2023), 85–111.

[7] Stefanie Betz and Birgit Penzenstadler. 2025. With great power comes great responsibility: The role of software engineers. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–21.

[8] Anastasiia Birillo, Mariia Tigina, Zarina Kurbatova, Anna Potriasaeva, Ilya Vlasov, Valerii Ovchinnikov, and Igor Gerasimov. 2024. Bridging education and development: Ides as interactive learning platforms. In *Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments*. 53–58.

[9] Yuriy Brun and Alexandra Meliou. 2018. Software fairness. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 754–759.

[10] Liyi Cai, Yijie Ren, Yitong Zhang, and Jia Li. 2025. AI-Driven Self-Evolving Software: A Promising Path Toward Software Automation. *arXiv preprint arXiv:2510.00591* (2025).

[11] Srividhya Chandrasekaran. 2024. Enhancing Developer Experience by Reducing Cognitive Load: A Focus on Minimization Strategies. *International Journal of Computer Trends and Technology* 72, 1 (2024), 99–103.

[12] Valerie Chen, Ameet Talwalkar, Robert Brennan, and Graham Neubig. 2025. Code with me or for me? how increasing ai automation transforms developer workflows. *arXiv preprint arXiv:2507.08149* (2025).

[13] Mariana Coutinho, Lorena Marques, Anderson Santos, Marcio Dahia, Cesar França, and Ronnie de Souza Santos. 2024. 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*. 131–138.

[14] Yinlin Deng, Chunqiu Steven Xia, Chenyuan Yang, Shizhuo Dylan Zhang, Shujing Yang, and Lingming Zhang. 2024. 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* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 70, 13 pages. https://doi.org/10.1145/3597503.3623343

[15] Gordana Dodig-Crnkovic, Gianfranco Basti, and Tobias Holstein. 2025. Delegating Responsibilities to Intelligent Autonomous Systems: Challenges and Benefits. *Journal of Bioethical Inquiry* (2025), 1–8.

[16] Kasra Ferdowsi, Ruanqianqian Huang, Michael B James, Nadia Polikarpova, and Sorin Lerner. 2024. Validating AI-generated code with live programming. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–8.

[17] 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 Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–30.

[18] Lucian Gonçales, Kleinner Farias, Bruno da Silva, and Jonathan Fessler. 2019. Measuring the cognitive load of software developers: A systematic mapping study. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE, 42–52.

[19] Ahmed E Hassan, Hao Li, Dayi Lin, Bram Adams, Tse-Hsun Chen, Yutaro Kashiwa, and Dong Qiu. 2025. Agentic Software Engineering: Foundational Pillars and a Research Roadmap. *arXiv preprint arXiv:2509.06216* (2025).

[20] Ahmed E Hassan, Gustavo A Oliva, Dayi Lin, Boyuan Chen, Zhen Ming, et al. 2024. Towards AI-native software engineering (SE 3.0): A vision and a challenge roadmap. *arXiv preprint arXiv:2410.06107* (2024).

[21] Brian Houck, Travis Lowdermilk, Cody Beyer, Steven Clarke, and Ben Hanrahan. 2025. The SPACE of AI: Real-World Lessons on AI's Impact on Developers. *arXiv preprint arXiv:2508.00178* (2025).

[22] Sarah Inman, Ambar Murillo, Sarah D'Angelo, Adam Brown, and Collin Green. 2025. Seamful AI for Creative Software Engineering. *IEEE Software* (2025). https://doi.org/10.1109/MS.2025.3534085 Conference Name: IEEE Software.

[23] Eirini Kalliamvakou, Christian Bird, Thomas Zimmermann, Andrew Begel, Robert DeLine, and Daniel M German. 2017. What makes a great manager of software engineers? *IEEE Transactions on Software Engineering* 45, 1 (2017), 87–106.

[24] Raula Gaikovina Kula and Christoph Treude. 2025. The Shift from Writing to Pruning Software: A Bonsai-Inspired IDE for Reshaping AI Generated Code.

[25] Aayush Kumar, Yasharth Bajpai, Sumit Gulwani, Gustavo Soares, and Emerson Murphy-Hill. 2025. How Developers Use AI Agents: When They Work, When They Don't, and Why. *arXiv preprint arXiv:2506.12347* (2025).

[26] Mark Marron. 2024. A New Generation of Intelligent Development Environments. In *Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments*. 43–46.

[27] Edward Meade, Emma O'Keeffe, Niall Lyons, Dean Lynch, Murat Yilmaz, Ulas Gulec, Rory V O'Connor, and Paul M Clarke. 2019. The changing role of the software engineer. In *European conference on software process improvement*. Springer, 682–694.

[28] Ahmed Menshawy, Zeeshan Nawaz, and Mahmoud Fahmy. 2024. Navigating challenges and technical debt in large language models deployment. In *Proceedings of the 4th Workshop on Machine Learning and Systems*. 192–199.

[29] Anh Nguyen-Duc, Beatriz Cabrero-Daniel, Adam Przybylek, Chetan Arora, Dron Khanna, Tomas Herda, Usman Rafiq, Jorge Melegati, Eduardo Guerra, Kai-Kristian Kemell, et al. 2025. Generative artificial intelligence for software engineering—A research agenda. *Software: Practice and Experience* (2025).

[30] Ipek Ozkaya. 2023. Application of large language models to software engineering tasks: Opportunities, risks, and implications. *IEEE Software* 40, 3 (2023), 4–8.

[31] Ipek Ozkaya. 2023. The next frontier in software development: AI-augmented software development processes. *IEEE Software* 40, 4 (2023), 4–9.

[32] Shidong Pan, Litian Wang, Tianyi Zhang, Zhenchang Xing, Yanjie Zhao, Qinghua Lu, and Xiaoyu Sun. 2024. " 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).

[33] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2025. Asleep at the keyboard? assessing the security of github copilot's code contributions. *Commun. ACM* 68, 2 (2025), 96–105.

[34] 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 Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–17.

[35] Zeeshan Rasheed, Muhammad Waseem, Malik Abdul Sami, Kai-Kristian Kemell, Aakash Ahmad, Anh Nguyen Duc, Kari Systä, and Pekka Abrahamsson. 2024. Autonomous agents in software development: A vision paper. In *International Conference on Agile Software Development*. Springer Nature Switzerland Cham, 15–23.

[36] Ruksit Rojpaisarnkit, Youmei Fan, Kenichi Matsumoto, and Raula Gaikovina Kula. 2025. How Natural Language Proficiency Shapes GenAI Code for Software Engineering Tasks. *IEEE Software* (2025), 1–7. https://doi.org/10.1109/MS.2025.3622690

[37] Mansour Sami, Ashkan Sami, and Pete Barclay. 2023. 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)*. IEEE, 391–396.

[38] Agnia Sergeyuk, Ekaterina Koshchenko, Ilya Zakharov, Timofey Bryksin, and Maliheh Izadi. 2024. The Design Space of in-IDE Human-AI Experience. *arXiv preprint arXiv:2410.08676* (2024).

[39] Agnia Sergeyuk, Ilya Zakharov, Ekaterina Koshchenko, and Maliheh Izadi. 2025. Human-AI Experience in Integrated Development Environments: A Systematic Literature Review. *arXiv preprint arXiv:2503.06195* (2025).

[40] Tasha Settewong, Youmei Fan, Raula Gaikovina Kula, and Kenichi Matsumoto. 2025. Human to Document, AI to Code: Three Case Studies of Comparing GenAI for Notebook Competitions.

[41] Wannita Takerngsaksiri, Jirat Pasuksmit, Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Ruixiong Zhang, Fan Jiang, Jing Li, Evan Cook, Kun Chen, and Ming Wu. 2025. Human-in-the-loop software development agents. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 342–352.

[42] Christoph Treude and Marco A Gerosa. 2025. 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)*. IEEE, 236–240.

[43] Christoph Treude and Hideaki Hata. 2023. 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)*. IEEE, 624–629.

[44] Christoph Treude and Raula Gaikovina Kula. 2025. Interacting with AI Reasoning Models: Harnessing "Thoughts" for AI-Driven Software Engineering. arXiv:2503.00483

[45] Christoph Treude and Margaret-Anne Storey. 2025. Generative AI and Empirical Software Engineering: A Paradigm Shift. *arXiv preprint arXiv:2502.08108* (2025).

[46] Michele Tufano, Anisha Agarwal, Jinu Jang, Roshanak Zilouchian Moghaddam, and Neel Sundaresan. 2024. AutoDev: Automated AI-driven development. *arXiv preprint arXiv:2403.08299* (2024).

[47] Rasmus Ulfsnes, Nils Brede Moe, Viktoria Stray, and Marianne Skarpen. 2024. Transforming software development with generative ai: Empirical insights on collaboration and workflow. In *Generative AI for effective software development*. Springer, 219–234.

[48] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. 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*. 1–7.

[49] Qiaosi Wang, Koustuv Saha, Eric Gregori, David Joyner, and Ashok Goel. 2021. 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*. 1–14.

[50] Ruotong Wang, Ruijia Cheng, Denae Ford, and Thomas Zimmermann. 2024. Investigating and designing for trust in ai-powered code generation tools. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*. 1475–1493.

[51] Danny Weyns, Thomas Bäck, Rene Vidal, Xin Yao, and Ahmed Nabil Belbachir. 2023. The vision of self-evolving computing systems. *Journal of Integrated Design and Process Science* 26, 3-4 (2023), 351–367.

[52] Chunqiu Steven Xia, Yuxiang Wei, and Lingming Zhang. 2023. Automated Program Repair in the Era of Large Pre-trained Language Models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1482–1494. https://doi.org/10.1109/ICSE48619.2023.00129

[53] Liang Yu. 2025. Paradigm shift on Coding Productivity Using GenAI. *arXiv preprint arXiv:2504.18404* (2025).

[54] Zhiqiang Yuan, Mingwei Liu, Shiji Ding, Kaixin Wang, Yixuan Chen, Xin Peng, and Yiling Lou. 2024. Evaluating and Improving ChatGPT for Unit Test Generation. *Proc. ACM Softw. Eng.* 1, FSE, Article 76 (July 2024), 24 pages. https://doi.org/10.1145/3660783

[55] Shao Zhang, Xihuai Wang, Wenhao Zhang, Yongshan Chen, Landi Gao, Dakuo Wang, Weinan Zhang, Xinbing Wang, and Ying Wen. 2024. 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).

[56] Xin Zhang and Victor S Sheng. 2024. Bridging the gap: representation spaces in neuro-symbolic AI. *arXiv preprint arXiv:2411.04393* (2024).