Exploring the Impact of Artificial Intelligence on Complex Software System Developers' Work: A Self-Determination Theory Perspective

Friedrich Schadow
HCI and Accessibility
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
friedrich.schadow@kit.edu

Kathrin Gerling
HCI and Accessibility
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
kathrin.gerling@kit.edu

Felix Kretzer
human-centered systems lab (h-lab)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
felix.kretzer@kit.edu

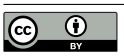
Alexander Maedche
human-centered systems lab (h-lab)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
alexander.maedche@kit.edu

Abstract

Artificial intelligence is rapidly adopted in software development resulting in a transformation of developers' work. This is particularly important in the context of complex software systems, such as enterprise software or cyber-physical systems, where development involves interconnected codebases and collaboration among multiple actors with deep domain knowledge. However, little is known about how AI affects developers working on complex software systems. In this paper, we examine how AI influences complex software system developers' work experiences through the lens of Self-Determination Theory (SDT). We report findings from a qualitative interview study with 14 developers involved in complex software system development that focused on consistency management and conflict resolution. Our results show that AI use is perceived to enhance developers' capabilities, while simultaneously increasing expectations regarding speed and quality of implementation. However, these benefits are accompanied by a persistent need for human oversight to ensure consistency and alignment with complex software systems. Furthermore, AI introduces tensions within development teams which have implications for consistency management. We discuss future research opportunities for the design of AI-based tools that support developers' self-determination and how to support the basic psychological needs of autonomy, competence and relatedness in complex software system development environments.

CCS Concepts

• **Human-centered computing** → **Empirical studies in HCI**; **HCI theory, concepts and models**; • **Software and its engineering** → **Collaboration in software development**.



CHIWORK Adjunct '26, Linz, Austria

ACM ISBN 979-8-4007-2598-2/26/06
https://doi.org/10.1145/3805029.3818286

Keywords

Software Development, Developer Experience, Human-AI Collaboration, Complex Systems, Self-Determination Theory

ACM Reference Format:
Friedrich Schadow, Kathrin Gerling, Felix Kretzer, and Alexander Maedche. 2026. Exploring the Impact of Artificial Intelligence on Complex Software System Developers' Work: A Self-Determination Theory Perspective. In *Adjunct Proceedings of the 5th Annual Symposium on Human-Computer Interaction for Work (CHIWORK Adjunct '26), June 22–25, 2026, Linz, Austria.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3805029.3818286

1 Introduction and Background

Artificial Intelligence (AI)-based tools such as large language model (LLM)–based coding assistants are increasingly integrated into software development [8]. AI-based tools that support user interface design, code generation, debugging, and documentation are more and more embedded into developers' daily workflows. As a result, AI is quickly becoming a central component of modern software development practice. Recent studies and industry reports suggest that these systems can significantly increase productivity [29, 31]. At the same time, it has been recognized that AI can alter how developers interact with other human actors and tools [10, 15, 25].

Most existing research and discussions on the impact of AI-based tools on software development focus on relatively isolated development tasks, such as small-scale programming activities. However, in industry many developers work on complex software systems. We understand a complex software system as one that is large in scale and in the number of contributing teams [32], long-lived in that it must keep evolving to remain useful [17], and highly interconnected through dense technical and organizational dependencies among artifacts, components, and developers [6, 20], a complexity that is, in Brooks' terms, essential rather than accidental [4]. Examples include enterprise software [3] and automotive cyber-physical systems [5, 16]. In such complex environments, developers must continuously coordinate their work with interdependent artifacts, tools, stakeholders, and constraints [12]. A central aspect of this type of work is consistency management [26, 27], which can involve ensuring alignment between requirements, models, code, tests, and

documentation. Since AI-based tools often generate and modify exactly these interdependent artifacts, they intervene directly in consistency management, both helping to restore consistency [14, 15] and potentially introducing new inconsistencies must be reconciled. Since this work is often collaborative, knowledge-intensive and shaped by domain- and organization-specific requirements, the use of AI-based tools might not only influence the efficiency of individual tasks but also developers' understanding of the system, decision-making, and collaboration. Consequently, the impact of AI-based tools in complex software system development is likely to differ from that in smaller or more isolated development settings. For example, while AI-generated code may accelerate certain tasks, it may also reshape developers' roles, their sense of control over the development process, and how expertise is distributed within teams. Understanding these broader impacts is essential as AI-based tools become increasingly embedded in professional complex software system development environments.

Recent work by Wong et al. [30] has highlighted the potential of *Self-Determination Theory (SDT)* [22] - a theoretical framework for understanding how autonomy, competence, and relatedness shape motivation and well-being - as a lens to better understand the human dimension of software development in general. SDT is a macro-theory of human motivation that examines the biological, social, and cultural conditions that support or undermine individual well-being, based on the inherent human capacity for psychological growth and engagement [23]. It comprises a set of mini-theories that contribute individual perspectives on human motivation, with *Basic Psychological Need Theory (BPNT)* being the most prominent one, postulating that each human being has an innate need to experience autonomy (i.e., experiencing volition and control over one's actions), relatedness (i.e., feeling connected to and supported by others), and competence (i.e., feeling effective and capable in one's tasks) [23]. Within the *Human-Computer Interaction (HCI)* research community, SDT in general and BPNT in particular is routinely applied to understand interactions with computing systems, e.g., in the context of games [28], user experience design [21], persuasive technologies [19] and most recently, as a lens on software engineering [30]. We argue that this provides a useful lens for understanding how new technologies such as AI influences individuals' experiences of work and motivation. By examining the impact of AI through the three basic psychological needs of autonomy, competence, and relatedness, we take a first step toward informing the design of AI-based tools that support complex software system developers' self-determination at work.

In this paper, we raise the following **research question (RQ)**: *How does AI impact the workplace experience of complex software system developers, and what are implications of AI use for the satisfaction of the three basic human needs to experience competence, autonomy, and relatedness?* We address this research question through an exploratory qualitative study in which we carried out interviews with N=14 software developers that focused on consistency management processes, tool usage for consistency management and the use of AI-based tools and AI agents. Through Reflexive Thematic Analysis [1, 2], we craft three themes: (1) The perception of performance enhancement through AI-based tools, (2) the need for constant quality control when leveraging AI for consistency management, and (3) tensions that arise in this context.

Through our work, we contribute an initial understanding of challenges and opportunities for the use of AI in complex software development, implications for developer competence, autonomy, and relatedness, and lay the foundation for future research that explores AI-based tool development for developer workplaces from the perspective of SDT [23].

2 Methodology

To address our research question, we carry out semi-structured interviews. Below, we detail our methodology: we first give an overview of how SDT has been applied as our theoretical lens to inform the interview guide, we then present the recruiting process and participant characteristics, describe our study procedure, and finally we present our approach to data analysis.

2.1 Interview Guide

Our interview guide is structured into four parts, socio-demographic information, consistency management processes within the company, tool usage for consistency management and a part specifically on the usage of AI tools and agents. Reflecting SDT, we crafted individual questions, e.g., *"How much scope for action and decision-making did you have in this process?"* (**Autonomy**), "*How well informed and capable did you feel at that moment* (a specific consistency conflict) *regarding the conflict and possible solutions?*" (**Competence**) and *How was coordination between the involved parties organized?* (**Relatedness**). With respect to the basic psychological needs, these questions sought to examine how the basic psychological needs of autonomy (i.e., experiencing volition and control over one's actions), the need to experience competence (i.e., feeling effective and capable in one's tasks), and to experience relatedness (i.e., feeling connected to and supported by others) are supported or thwarted as a result of AI tool use.

2.2 Participants and Procedure

We recruited 14 participants via social media, word of mouth, and a panel platform from our university that seeks to involve the local community in research. 14 persons (1 women, 13 men and 0 non-binary persons; average age 37.92, SD=11.98) took part in the interview study (see Appendix A for an overview). Work experience ranged from 3 to 50 years, with participants working in the development of cyber-physical systems and complex software systems as software engineers (both in intermediate and senior roles), technical engineers or project managers (having previously worked in software development). 13 participants had previous experience in the use of AI-based tools in their own development practice, whereas 1 person only had indirect experiences therewith as a results of colleagues using AI. A total of 4 participants entirely rejected the use of AI in their work. Participants with a critical stance to AI were included to reflect the heterogenous reception of AI tools among developers. On average, interviews lasted about 45 minutes; all interviews were audio-recorded via the institution-internally hosted video conference tool *Jitsi* and later on transcribed using an internally hosted AI transcription tool based on *Whisper*, in line with local data protection practice. The research protocol was approved by the ethics committee of our university.

2.3 Data Analysis

Data were analyzed applying *Reflexive Thematic Analysis (TA)* [1, 2], following the protocol devised by Braun & Clarke [2]. This approach is well-suited to understanding lived experience, and adopting an inductive coding approach is in line with the exploratory nature of our work that seeks to develop a bottom-up understanding of developers' experiences. In the first step, the first author read and re-read transcripts, familiarizing themselves with the data. Then, the first author generated initial codes, which were iteratively refined and grouped into preliminary themes. For example, the statement *"With AI tools, you can work much faster and more effectively, no matter what you're doing."* was coded as Peceived Process Acceleration, and later collated into Theme 1: AI enhances perceived performance, expands individual capabilities, but raises expectations. Codes and themes were reviewed within the research team, with the final themes focusing on the performance enhancing effects of AI tools and the resulting rise of expectations, the perceived need for constant human review of AI-generated results, and tension created through AI usage regarding trust and reliability of its outputs. Considering the subjective nature of reflexive TA [2], we briefly want to reflect on our positionality [11]. We are a team of researchers with backgrounds in HCI, Information Systems, and Psychology. Some team members have previously worked in industry in roles interfacing with software development; some team members previously or currently teach courses focusing on programming and software engineering. We believe that AI tools can be a helpful resource for developers, but that these must be designed in a way mindful of human preferences.

3 Results

In the following section, we present our three main themes: Performance enhancing factors of AI tools, the need for human review of AI results and distrust in the capabilities of AI in complex software system environments.

3.1 Theme 1: AI Use Enhances Perceived Performance and Individual Capabilities, But Raises Expectations

Participants working on complex software systems who used AI-based tools generally highlighted a perceived acceleration of work performance, which was also associated with shifting expectations regarding the speed and quality at which solutions need to be implemented. Working with AI agents was frequently reported to reduce time spent on routine tasks and to enable rapid iteration of possible solutions, especially in complex environments, as one of the developers stated: *"When working with a monolithic architecture, it's also helpful to ask AI tools questions about the code. I like to use it, for example, when I want to know which paths I can take to reach a specific part of the code"* (P11). Developers also noted that they would expect others to deliver solutions more quickly and of higher quality, while simultaneously experiencing increased expectations towards their own work performance due to the availability of AI tools, e.g., *"Today, after just a day or two, people are wondering why it's not done yet, even though I'm actually just a prompt away from getting it done"* (P4). This shift is partly rooted in the effect of AI tools helping developers engage with tasks beyond their immediate area of expertise, by expanding existing skills into new areas and compensating for missing skills, or knowledge, thereby supporting cross-domain work, as os participant said: *"Especially when I'm working in a language that isn't my preferred one, this helps immensely. When you have a sort of pseudocode and think it should look something like that, and then the IDE already underlines everything in red, and you can select it and say, Convert this to JavaScript or TypeScript"* (P10). For some participants, this perceived increase in autonomy did however lead to reduced social interaction, as they relied less on colleagues for initial problem-solving, stating that *"It means I bother my colleagues less with silly questions, because I can just brainstorm with AI. It makes my life easier and lonelier"* (P1). Here, the interviewed developers also noted that the increased process acceleration can introduce challenges to consistency management, as solutions generated with external AI tools may not align with existing system structures. The need for human-in-the-loop practices was repeatedly emphasized as essential to ensure coherent and maintainable implementations.

3.2 Theme 2: AI Use Requires Human Expertise and Constant Reviewing

Despite the perceived benefits for development speed and quality, participants consistently emphasized the need for human expertise and process knowledge as well as ongoing human review of AI-generated output, which was particularly relevant for consistency management. One participant stated *"There was a case before where AI was used, and there was an error in it that wasn't caught, and then someone commented on it, and it turned out, yeah, that was the generated code. Yeah, so when you notice that someone basically just asked the AI in a comment or just threw it in there, so to speak. So for me, it would be really important that people actually take a close look at what the AI generates and don't just accept it at face value"* (P13). Some developers also mentioned that guidelines for AI-usage are not only self-imposed, but also enforced by company policies that were introduced as a consequence of the emergence of AI tools. Such AI guidelines include breaking down AI-generated solutions into smaller components to enable more effective reviews and reduce the risk of unintended errors and redundancies. An example provided states: *"One recommendation that we have in the (AI) guideline is that if you are using AI-assisted tooling to make sure that the number of changes or the number of diffs are not too high"* (P2). Based on the constant need for human control loops, some developers emphasized that they would not be entirely replaced by AI, but that their roles would shift towards supervision and validation of AI-generated results: *"I think that these standards of quality and accuracy are just difficult to teach to AI, and that humans will continue to have a better eye for these things for a long time to come"* (P10). However, developers highlighted that the development of complex systems and consistency assurance remains a fundamentally human task that requires domain expertise and an understanding of broader system contexts. Participants further expressed concerns that isolated AI-generated solutions may negatively impact the overall quality of implementation and undermine the competence gain of developers, a tension we explore in the following section.

3.3 Theme 3: AI Use Creates Tensions Related to Trust, Effectiveness, and Expertise

Participants who used AI in their professional work and those that decided against it expressed a variety of concerns that extended beyond consistency management. Senior developers in particular felt that AI tools can lead to additional workload due to flawed or incomplete outputs when working on cyber-physical systems, with one AI critical participant stating: *"A very representative example was when we visualized complex multi-modal data from different physiological sensors and cameras. When an LLM was used to generate code for plotting and visualizing the data, it used some magic numbers to decode the video packets that actually made it look like the data itself was scrambled or desynchronized. Of course, that's not the case. It was a bug that was introduced in the visualization process by the LLM itself, which was a big point of concern"* (P6). Some participants expressed a sense of distrust towards AI tools and deliberately avoided their use, especially in contexts where maintaining a coherent system structure was crucial. This distrust was reinforced by experiences with AI hallucinations, which were described as hard to identify without sufficient expertise, further emphasizing the aforementioned need for human review. Developers with a higher level of seniority also highlighted that AI usage among junior developers may negatively impact their skill development by encouraging reliance on isolated solutions and limiting their understanding of broader system structures and best practices in software development, making consistency management more challenging, e.g., *"I think it actually causes brain rot. It is a very harsh comment, but it's something that I see quite a lot, and also across the students that we have in the university, that the more LLM usage there is, the less competent people are in developing complex cyber-physical systems"* (P6). Participants further noted that this reliance may inhibit learning and skill maintenance, as it was seen to reduce engagement with underlying systems and documentation, resulting in a gradual loss of process knowledge over time. This risk was particularly emphasized by senior developers, including both participants with direct experience using AI-based tools in their own development practice and participants who were more critical of AI.

4 Discussion

In this section, we first answer our research question. Then, we discuss opportunities for future work with focus on AI tools, complex software systems development, and developer self-determination.

4.1 Answering the Research Question

We articulate the research question of how AI impacts the workplace experience of software developers, and what are implications of AI use for the satisfaction of the three basic human needs to experience competence, autonomy, and relatedness? Our results show that developers' workplace experiences were impacted by their own or their colleagues' use of AI tools. The strongest motivator for the use of AI-based tools was to close skill gaps (see Section 3.1), which is directly related to the human need to experience competence. More critical perspectives on AI use were related to a perceived lack of reliability of generated code (see Section 3.2), which increased challenges related to consistency management. Here, we conclude that the use of AI in the development of complex software systems affect relationships between developers of different seniority (see section 3.3), possibly reducing the experience of relatedness.

4.2 Opportunities for the Design of AI Tools to Support Complex Software System Developer Self-Determination

Based on our findings, we identified three key opportunities for the design of AI tools in complex software system development, acknowledging that developer self-determination needs to be supported alongside their productivity.

Opportunity 1: Designing AI Tools That Develop and Augment Complex Software System Developers' Skills and Competence at Different Career Stages. Many participants in our study applied AI to close skill gaps (see Section 3.1), but did so in the context of their own - often extensive - development experience, while AI use by junior developers was viewed more critically. Here, future work should draw upon the rich body of work in Computer Science Education (e.g., on scaffolding [7] and consequences of AI tool use on learning [18]) and explore AI tools that adapt to different career stages, providing scaffolding for junior developers in a way that still enables individual skill development, and examining how **tailored AI tools can contribute to and protect the human need to experience competence**.

Opportunity 2: Designing Trustworthy and Dependable AI Tools for Consistency Management in Complex Software System Development Ensuring Autonomy. Our results show that developers valued their autonomy in instances of consistency management, and did not exclusively rely on AI (see Section 3.3) as a result of a perceived lack of reliability. This suggests an opportunity for future work to contribute **transparent, explainable and hence dependable AI support for consistency management that supports developer autonomy**. For example, trust could be built via measures to increase transparency of when and where AI-generated code is introduced into code bases similar to existing efforts to attribute authorship [13] and reviewing status [24] in software development.

Opportunity 3: Applying AI Tools With Caution to Protect Meaningful Experiences of Relatedness in Complex Software System Development. The use of AI tools may reduce exchange among developers (see Section 3.1) and introduce negative interactions (see Section 3.3). In consistency management and conflict resolution, this is particularly relevant as the importance of constructive human negotiation has long been recognized [9]. Future work should explore how to **augment code generation tools with opportunity for meaningful exchange within teams, explicitly designing for relatedness**. For example, such tools could turn generated suggestions into shared review objects that make the affected requirement, linked artifacts, and the AI's rationale visible to multiple roles, thereby inviting brief negotiation instead of silent acceptance or ex-post conflict resolution.

4.3 Limitations and Conclusion

Our work needs to be interpreted in the light of the exploratory and qualitative nature of our research approach. We only include data

from a small number of participants located in Western Europe, but discuss these in depth. Future work should follow up with a larger sample to achieve generalizability of results, and take into account additional factors such as developer roles, seniority, domains, and workplace practices in other cultures. Overall, our work contributes to the growing body of research that highlights the tensions in software development practice and workplace experiences in the light of increased use of AI tools. Specifically, our study of the application of AI in the development of complex software systems is a starting point for the further exploration of how tailored AI tools can contribute to software developers' self-determination.

Acknowledgments

The authors acknowledge the use of *Grammarly* and large language models for language and layout refinement, and a local version of *Whisper* for interview transcription. While these tools supported paper preparation, all research findings, interpretations, and contributions remain the authors' original work.

References

[1] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. doi:10.1191/1478088706qp063oa

[2] Virginia Braun and Victoria Clarke. 2019. Reflecting on reflexive thematic analysis. *Qualitative Research in Sport, Exercise and Health* 11, 4 (2019), 589–597. doi:10.1080/2159676X.2019.1628806

[3] L. Brehm, A. Heinzl, and M.L. Markus. 2001. Tailoring ERP systems: a spectrum of choices and their implications. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*. IEEE, New York, NY, USA, 9 pp.–. doi:10.1109/HICSS.2001.927130

[4] Brooks. 1987. No Silver Bullet Essence and Accidents of Software Engineering. *Computer* 20, 4 (1987), 10–19. doi:10.1109/MC.1987.1663532

[5] Manfred Broy. 2006. Challenges in automotive software engineering. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. Association for Computing Machinery, New York, NY, USA, 33–42. doi:10.1145/1134285.1134292

[6] Marcelo Cataldo, Audris Mockus, Jeffrey A. Roberts, and James D. Herbsleb. 2009. Software Dependencies, Work Dependencies, and Their Impact on Failures. *IEEE Trans. Softw. Eng.* 35, 6 (Nov. 2009), 864–878. doi:10.1109/TSE.2009.42

[7] Dengkang Chen, Yi Zhang, Heng Luo, Zhimin Gao, Lufang Yu, and Yuru Lin. 2026. Exploring the Impact of Scaffolding in Programming on Students' Computational Thinking: Evidence From a Three-Level Meta-Analysis. *Journal of Educational Computing Research* 64, 1 (Jan. 2026), 208–237. doi:10.1177/07356331251386618

[8] Nicole Davila, Igor Wiese, Igor Steinmacher, Lucas Lucio da Silva, Andre Kawamoto, Gilson Jose Peres Favaro, and Ingrid Nunes. 2024. An Industry Case Study on Adoption of AI-based Programming Assistants. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '24)*. Association for Computing Machinery, New York, NY, USA, 92–102. doi:10.1145/3639477.3643648

[9] Prasun Dewan and Rajesh Hegde. 2007. Semi-synchronous conflict detection and resolution in asynchronous software development. In *ECSCW 2007*, Liam J. Bannon, Ina Wagner, Carl Gutwin, Richard H. R. Harper, and Kjeld Schmidt (Eds.). Springer London, London, 159–178. doi:10.1007/978-1-84800-031-5_9

[10] Werner Geyer, Jessica He, Daita Sarkar, Michelle Brachman, Chris Hammond, Jennifer Heins, Zahra Ashktorab, Carlos Rosemberg, and Charlie Hill. 2025. A Case Study Investigating the Role of Generative AI in Quality Evaluations of Epics in Agile Software Development. In *Proceedings of the 4th Annual Symposium on Human-Computer Interaction for Work (CHIWORK '25)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3729176.3729200

[11] Prashneel Ravisan Goundar. 2025. Researcher Positionality: Ways to Include it in a Qualitative Research Design. *International Journal of Qualitative Methods* 24 (2025), 16094069251321251. doi:10.1177/16094069251321251

[12] James D. Herbsleb and Rebecca E. Grinter. 1999. Splitting the organization and integrating the code: Conway's law revisited. In *Proceedings of the 21st International Conference on Software Engineering (ICSE '99)*. Association for Computing Machinery, New York, NY, USA, 85–95. doi:10.1145/302405.302455

[13] Vaibhavi Kalgutkar, Ratinder Kaur, Hugo Gonzalez, Natalia Stakhanova, and Alina Matyukhina. 2020. Code Authorship Attribution: Methods and Challenges. *Comput. Surveys* 52, 1 (Jan. 2020), 1–36. doi:10.1145/3292577

[14] Kristian Kolthoff, Felix Kretzer, Christian Bartelt, Alexander Maedche, and Simone Paolo Ponzetto. 2024. Interlinking User Stories and GUI Prototyping: A Semi-Automatic LLM-Based Approach. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, Reykjavik, Iceland, 380–388. doi:10.1109/RE59067.2024.00045

[15] Felix Kretzer, Kristian Kolthoff, Christian Bartelt, Simone Paolo Ponzetto, and Alexander Maedche. 2025. Closing the Loop between User Stories and GUI Prototypes: An LLM-Based Assistant for Cross-Functional Integration in Software Development.. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–19. doi:10.1145/3706598.3713932

[16] Edward A. Lee. 2008. Cyber Physical Systems: Design Challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, New York, NY, USA, 363–369. doi:10.1109/ISORC.2008.25

[17] M.M. Lehman. 1980. Programs, life cycles, and laws of software evolution. *Proc. IEEE* 68, 9 (1980), 1060–1076. doi:10.1109/PROC.1980.11805

[18] Dandan Liu, Guangrui Fan, and Lihu Pan. 2026. Tool, tutor, or crutch?: A grounded theory of cognitive scaffolding and offloading in AI-assisted programming education. *International Journal of STEM Education* 13, 1 (March 2026), 10. doi:10.1186/s40594-025-00592-w

[19] Fidelia A. Orji, Francisco J. Gutierrez, and Julita Vassileva. 2026. Motivation profiles: understanding interplay of persuasive strategies and self-determination theory. *Behaviour & Information Technology* 45, 8 (May 2026), 1567–1586. doi:10.1080/0144929X.2025.2522200

[20] D. L. Parnas. 1972. On the criteria to be used in decomposing systems into modules. *Commun. ACM* 15, 12 (Dec. 1972), 1053–1058. doi:10.1145/361598.361623

[21] Dorian Peters, Rafael A. Calvo, and Richard M. Ryan. 2018. Designing for Motivation, Engagement and Wellbeing in Digital Experience. *Frontiers in Psychology* 9 (May 2018), 797. doi:10.3389/fpsyg.2018.00797

[22] Richard M. Ryan and Edward L. Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist* 55, 1 (2000), 68–78. doi:10.1037/0003-066X.55.1.68

[23] Richard M. Ryan and Maarten Vansteenkiste. 2023. Self-Determination Theory: Metatheory, Methods, and Meaning. In *The Oxford Handbook of Self-Determination Theory* (1 ed.), Richard M. Ryan (Ed.). Oxford University Press, Oxford, 3–30. doi:10.1093/oxfordhb/9780197600047.013.2

[24] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern code review: a case study at google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '18)*. Association for Computing Machinery, New York, NY, USA, 181–190. doi:10.1145/3183519.3183525

[25] Viktoria Stray, Astri Barbala, and Viggo Tellefsen Wivestad. 2025. Human-AI Collaboration in Software Development: A Mixed-Methods Study of Developers' Use of GitHub Copilot and ChatGPT. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25)*. Association for Computing Machinery, New York, NY, USA, 1325–1332. doi:10.1145/3696630.3730566

[26] P. Tarr and L.A. Clarke. 1998. Consistency management for complex applications. In *Proceedings of the 20th International Conference on Software Engineering*. IEEE, New York, NY, USA, 230–239. doi:10.1109/ICSE.1998.671122

[27] Michael Alexander Tröls, Luciano Marchezan, Atif Mashkoor, and Alexander Egyed. 2022. Instant and global consistency checking during collaborative engineering. *Software and Systems Modeling* 21, 6 (Dec. 2022), 2489–2515. doi:10.1007/s10270-022-00984-4

[28] April Tyack and Elisa D. Mekler. 2020. Self-Determination Theory in HCI Games Research: Current Uses and Open Questions. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu HI USA, 1–22. doi:10.1145/3313831.3376723

[29] Thomas Weber, Maximilian Brandmaier, Albrecht Schmidt, and Sven Mayer. 2024. Significant Productivity Gains through Programming with Large Language Models. *Proc. ACM Hum.-Comput. Interact.* 8, EICS (June 2024), 1–29. doi:10.1145/3661145

[30] Novia Wong, Nai-Yu Cheng, Bruna Oewel, Katherine E Genuario, SarahElizabeth Stoeckl, Stephen M. Schueller, Iftekhar Ahmed, André Van Der Hoek, and Madhu Reddy. 2025. 'It's a spectrum': Exploring Autonomy, Competence, and Relatedness in Software Development Processes and Tools. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–19. doi:10.1145/3706598.3713250

[31] Albert Ziegler, Eirini Kalliamvakou, X. Alice Li, Andrew Rice, Devon Rifkin, Shawn Simister, Ganesh Sittampalam, and Edward Aftandilian. 2022. Productivity assessment of neural code completion. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming (MAPS 2022)*. Association for Computing Machinery, New York, NY, USA, 21–29. doi:10.1145/3520312.3534864

[32] Darja Šmite, Nils Brede Moe, Aivars Šāblis, and Claes Wohlin. 2017. Software teams and their knowledge networks in large-scale software development. *Information and Software Technology* 86 (2017), 71–86. doi:10.1016/j.infsof.2017.01.003

A Participant Overview

Table 1: Participant Overview

Participant	Experience in Years	Personnel Responsibility
P1	10	no
P2	8	no
P3	6	no
P4	12	yes
P5	29	yes
P6	4	yes
P7	13	no
P8	6	no
P9	4	no
P10	7	no
P11	3	no
P12	50	yes
P13	5	no
P14	17	no