


See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/378567262>

Transforming Software Development with Generative AI: Empirical Insights on Collaboration and Workflow

Preprint · February 2024

CITATIONS
0

4 authors:




Rasmus Ulfnes

SINTEF

14 PUBLICATIONS 53 CITATIONS

SEE PROFILE




Viktoria Stray

University of Oslo

93 PUBLICATIONS 1,419 CITATIONS

SEE PROFILE

READS
1,511




Nils Brede Moe

SINTEF

187 PUBLICATIONS 6,351 CITATIONS

SEE PROFILE



Marianne Skarpen

Norwegian University of Science and Technology

2 PUBLICATIONS 2 CITATIONS

SEE PROFILE

Transforming Software Development with Generative AI: Empirical Insights on Collaboration and Workflow

Rasmus Ulfsnes¹[0000–1111–2222–3333], Nils Brede Moe¹[0000–0003–2669–0778],
Viktoria Stray^{1,2}[0000–0002–6032–2074], and Marianne
Skarpen³[0009–0005–5575–316X]

¹ SINTEF, Strindvegen 4, 7030 Trondheim, Norway
{`rasmus.ulfsnes`, `nils.b.moe`}@sintef.no

² Department of Informatics, University of Oslo, Oslo, Norway
`stray@uio.no`

³ NTNU, Trondheim, Norway
`marskarp@stud.ntnu.no`

Abstract. Generative AI (GenAI) has fundamentally changed how knowledge workers, such as software developers, solve tasks and collaborate to build software products. Introducing innovative tools like ChatGPT and Copilot has created new opportunities to assist and augment software developers across various problems. We conducted an empirical study involving interviews with 13 data scientists, managers, developers, designers, and frontend developers to investigate the usage of GenAI. Our study reveals that ChatGPT signifies a paradigm shift in the workflow of software developers. The technology empowers developers by enabling them to work more efficiently, speed up the learning process, and increase motivation by reducing tedious and repetitive tasks. Moreover, our results indicate a change in teamwork collaboration due to software engineers using GenAI for help instead of asking co-workers which impacts the learning loop in agile teams.

Keywords: Agile software development · Product development · Teamwork.

1 Introduction

There is a growing trend among companies to adopt digitalization and engage in digital transformation[36]. This transformation process requires technologies as software, data and artificial intelligence [5], forcing a shift in the use of strategic frameworks [30], and new ways of developing technology for highly skilled employees with intelligent technology [36].

Technology for assisting developers with writing code, particularly using Integrated Development Environments (IDEs) is not a new concept [26,17]. The task of autocompletion of code, and generating of test, and various other tasks has been particularly interesting for software, as natural language matches quite

well as the software code is hypothesized to be a natural language [16]. Subsequently this hypothesis has led to lots of research on Artificial Intelligence (AI) for software engineering [32,33]. With the introduction of Generative Artificial Intelligence (GenAI) - a type of artificial intelligence (AI), both the software development processes and tooling have started to change fast. Further, GenAI can revolutionize software development by automating repetitive tasks, improving code quality, enhancing collaboration, providing data-driven insights, and ultimately accelerating the development lifecycle [31,25]. There is a growing research into how to use Copilot [28], or Generative AI systems such as Chat GPT [37], and its capability to automate software engineering tasks [20]. However, good tooling is not enough.

In order for a company to succeed with software product development, well-working teams and good processes are key. Software engineering is a social activity that is focused on close cooperation and collaboration between all team members [21] and across teams in the organization [2]. Therefore, it is important to note that while AI has great potential, it also comes with challenges [1] such as ethical considerations, data privacy concerns, the need for skilled professionals to handle the technology within software teams, and a potential change in the team dynamics. However, research on team dynamics is lacking. In order to understand the effects on team-dynamics however we also need to consider the individual work practices, to grasp the effects on a team level. This paper explores how are software engineer work practices transformed, and potential impact on the transformation on collaboration.

We have interviewed 13 data scientists, managers, developers, designers, and frontend developers to investigate how they use GenAI technology and how their workday has changed. Finally, we discuss how this technology might affect software development teamwork.

2 Related work

2.1 Productivity and work satisfaction

Competing for talents requires a conscious effort to offer an attractive workplace [24]. Further, the ability to balance the need and nature of the workdays for different team members is directly related to the outcome of the product [35]. For software developers, Meyer et al. [22] outline a framework that describes what makes a good workday. There are three main factors: value creation, efficient use of time, and perception. Value creation is about whether or not the developers feel they are creating something, and the factor has six sub-factors. The second factor, efficient use of time, has two sub-factors, meeting expectations and the ability to work focused. In essence, the assessment of a workday being good or bad is largely influenced by the expectations for the day. For example, if one anticipates a day filled with meetings, the day can be considered good even if most of the time is spent in meetings. However, if one hopes for a day of focused work and the day is filled with meetings, the day is perceived as a bad workday. Coworker interruptions were specifically described as negatively

influencing developers' ability to focus or work as planned, although being able to help a co-worker was generally considered positive and rewarding. Lastly, perception is about how they perceive their own productivity.

Developer satisfaction and work productivity are related, therefore they need to be key considerations for software companies [15]. More productive developers may be more satisfied, and more satisfied developers may be more productive. Autonomy, being able to complete tasks, and technical skills all affect productivity. By introducing new technology like GenAI, a team member's productivity may be positively affected. At the same time work culture and team collaboration are important for job satisfaction. An increased reliance on tools like GenAI may enhance individual productivity while inadvertently reducing inter-team interactions, ultimately affecting long-term job satisfaction and collective productivity. Introducing GenAI in software teams is therefore a balancing act.

2.2 Software development, knowledge work and technology

Software engineering requires the input and consolidation of various information to produce code [26]. Furthermore, with the advent of DevOps with its increase in speed of delivering continuously [13], increased accessibility to third-party libraries and frameworks, the process of software engineering has shifted from being about understanding the computer and the programming language towards understanding how to compose relevant libraries and frameworks, with applicable testing.

Software development encompasses more than just programming and teamwork, it also involves actively seeking knowledge online and in knowledge management systems [12], conducting testing and code reviewing [14], and taking advantage of software such as Integrated Development Environments. IDEs has been a researched topic for quite some time [26] as well as how online resources [8,18] aid and enhance the development process, both in speed and quality across varying ranges of experience. Through the use of IDEs, software engineers have gotten access to capabilities for refactoring, debugging, source repositories, third-party plugins[26] and auto-completion of code [6]. In addition, developers need to browse through a plethora of different files in existing software solutions in order to get a grasp of how changes to the code need to be implemented [17].

More recently, better auto-completion methodologies and technologies have been introduced to provide more context-relevant suggestions using statistical methods [6], and the naturalness of software [16] is a great target for utilizing natural language processing and generative AI. Github Copilot has shown promising effects for assisting developers in writing code, assisting with test writing [25,4]. Early studies on knowledge work show that generative AI are able to disseminate knowledge previously shown as tacit[7] and dramatically increase both quality and production[10]. Both studies show that the effects are most noticeable for the lower skilled workers while higher skilled workers have a lower increase in production and quality.

2.3 Teams, knowledge sharing and performance

Software product development is done in teams [19], therefore, the success of software development depends significantly on team performance. Today the premise is that software teams should be autonomous or self-managed [27]. In their review, Dickinson and McIntyre [11] identified and defined seven core components of teamwork. Using these components and their relationships as a basis, they proposed the teamwork model that is used in this work. The model consists of a learning loop of the following basic teamwork components: communication, team orientation, team leadership, monitoring, feedback, backup, and coordination. Later Moe et al. [23] used this model to explain agile teamwork. The introduction of GenAI is likely to affect these teamwork components.

3 Research Method and Analysis

This study was conducted in the context of two research programs on software development processes, where several companies introduced Generative Artificial Intelligence (GenAI) in their product development process. GenAI, especially those built on LLMs, is a new phenomenon that has not been previously studied. Due to the uncertain nature of the phenomenon, we chose an exploratory multi-case study [38]. We selected our informants using snowball sampling [3] in Slack asking for subjects that used GenAI for a wide range of activities. As of the tools used, our studies found that ChatGPT and GitHub Copilot were the most common for code and text, while some reported that they used Midjourney and DALL-E 2 for image creation.

3.1 Data Collection and Analysis

We interviewed 13 people, as shown in Table 1. The informant group had a broad range of roles: data scientists, managers, developers, designers, and front-end developers. Based on a literature review, we developed a semi-structured interview guide. Questions included: *How do you use GenAI services?* and *Which effects do you get from using them?*. The interviews were done by the first, third, fourth, and fifth authors to spread out any subjective biases.

The analysis was divided into two cycles of coding as suggested by Saldana [29], with the third and first author conducting a combination of descriptive and initial coding [9] on the first eight interviews. Then the third and first authors had a discussion about the emerging categories and themes. This led to a revised interview guide. Finally, the last five interviews were conducted.

After the last round of interviews, the fourth author performed a descriptive coding of the remaining interviews. At the same time, the first author performed a second cycle [29] focused coding of all interviews. Then, the observed themes and categories were merged through mutual workshops and discussions with all authors.

Table 1. Data sources

ID	Role	Work experience in years
I1	Developer & Team lead	5
I2	Developer	5
I3	Technical Strategy Consultant & Director	30
I4	Tech Lead & Machine learning engineer	6
I5	Principal Engineer and Enterprise Architect	15
I6	Data Science Manager	6
I7	Developer	10
I8	CTO	25
I9	Director	30
I10	Developer	N/A
I11	Designer	13
I12	Designer	3
I13	Developer	3

4 What is Generative AI Used for in Software Development?

Software development consists of a number of activities, ranging across multiple roles in cross-functional teams. When using source repository systems or IDEs, the use case is often clear. However, the use of GenAI takes on a much more individual form. There are currently no standards or norms for how, when, and for what purpose you should apply GenAI to, and employees in software-intensive organizations are using it based on their own preferences. GenAI tools for a wide range of activities. See Table 2 for an overview of such activities.

The type of GenAI activities and utilization depends on individual preferences, the task to be solved, and the user’s role in the organization. Developers typically use GenAI when working with the source code, while managers use it for, e.g. organizing workshops or creating content for PowerPoint presentations.

4.1 Asking for assistance when stuck

When a person was stuck on a particular problem or did not know how to proceed, ChatGPT was used as an assistant or fellow team member, where interacting with it using chat could help a person solve complex problems or get increased progress. For non-technical problems, this could be a case of writer’s block, formulations, or when they are zoning out: *”For me, the main thing is to get unstuck, whether I am struggling with writer’s block or formulations, just by interacting with ChatGPT and getting an immediate response is something else.”*

This highlights that there is an effect of just having the chat window open, and getting feedback without interrupting others. Formulating the problem to ChatGPT made it easier to keep focus on the task, helped on the thought processes, and helped see the problem in a new light. Developers referred to such

Table 2. List of GenAI Activities

Activity	Description
Asking for assistance when stuck	When stuck on a particular task, GenAI can help getting out of the slump.
Learning	GenAI provide an interactive way of learning new things.
Creating a virtual environment for a product	By asking GenAI to provide an simulated/virtual environment to learn and test products.
PowerPoint and email writing	GenAI is useful for helping with writing text for email, powerpoint.
Non-technical boilerplate	Providing a boilerplate for how to get started with powerpoint, workshops.
Boilerplate code	GenAI can provide boilerplate code that acts as a skeleton for further development.
Working with existing code	GenAI are used for refactoring, adding small features, making code more robust, converting code, debugging, writing tests, Search Engine Optimization (SEO).

interaction with ChatGPT as rubber-ducking. The idea of rubber-ducking is to explain the problem one seeks to solve to an inanimate object (e.g. a rubber duck), in an attempt to achieve a deeper understanding of the problem and a potential solution through the process of explaining it to someone (or something) using natural language.

Using GenAI as a sparring partner was both faster than asking human colleagues and also took away the feeling of disturbing them in their work. Further, being able to formulate the question as you would to a *human* felt easier than the alternate Google search, where you need to consider the specific keywords, and what results they can give you.

4.2 Learning

GenAI provided more opportunities and avenues for engagement in learning compared to reading books, using Google, or watching videos. One approach was to engage interactively with the chat, assigning a role to the AI like "act as a tester", and then engaging with that persona to learn about testing. They could then ask that persona to explain a particular topic like they were doing testing for the first time.

"And then I continue. And I notice that I learn much faster this way. Because it's like having a personal tutor. Where you can ask yourself questions. And then I always have to double-check."

The same approach was also applied when working on code, where using new features, or when working on areas that the informants were not that knowledgeable about, a data scientist explained how to use the technology to learn more about programming: *"Almost as if I were asking someone much more skilled, like a developer in this case."*

4.3 Virtual environments

Creating a virtual environment for a product was used to develop software for a trading platform. They engaged with the Chat GPT and asked it to simulate that it was a stock exchange. They then provided the Chat GPT with information about which stocks could be traded at the exchange and asked it to simulate different trading scenarios. This provided a novel way for the informant to understand the intricacies of a stock exchange. Further, this means that the Chat GPT had the context for the particular trading platform the informants were interested in.

"And then I said, 'now I'm going to make an application out of this in such-and-such language.' And so it has the context for everything while I kept asking it further questions." This integrated approach to both understanding a domain, also then produced the relevant context that ChatGPT could use to generate relevant code.

4.4 Copywriting

Maybe not surprisingly, Chat GPT was used to assist in copywriting text, especially useful when integrated into the tool the informant was using, getting live feedback; this was particularly useful for persons that were not native or fluent in English or Norwegian. However, some experienced that GenAI was not very useful for email and text writing for two main reasons, there was a significant overhead in engaging sufficiently with ChatGPT to create emails, and the quality did not get better.

"I think I have asked it to write emails, but for me it is just faster to write it myself. The formulation was better though."

4.5 Boilerplating code and text

Getting started with a relatively novel coding project in any company requires quite a lot of boilerplate code; this type of code does not add functionality relevant to the business case but is required to get the project up and running with the necessary declarations and structures. Both back-end and front-end developers used ChatGPT to create tailored boilerplate code:

"If I have a task, to create a list of tricks with something, and thumbs up and thumbs down on each element, for instance. I often start by describing what I want to ChatGPT. Then, it writes the code for me." GenAI was also used for repetitive or tedious non-technical tasks. For example, managers and architects stated to use ChatGPT to consolidate text used for production of bid to customers. One manager explained, "A bid I would normally have spent a lot of time in writing, I only spent 20 minutes on. Previously I would have spent a lot of time, looking for previous bids, adapting it and merging it. It is terrible to say it out loud [laughing], as this kind of is in someway reducing the need for my work. This type of work is what the company pays me to do."

Another example is when you are building up technical specifications and technical architectures where the style of the text is quite consistent but the content varies between use cases. Or getting feedback on emails, and getting a headstart on the writing of the text.

4.6 Working with existing code

This was the most common activity among software developers. GenAI was used on many different tasks, ranging from refactoring or simplifying code, code review, translating code from one programming language to another, and simply explaining the code. Testing was also well-suited for GenAI utilization, given its repetitive nature, GenAI was used to create numerous tests for the code. The informants also noted that the generated tests sometimes accounted for scenarios and test cases that they themselves had not thought of.

"It was mostly a matter of thinking up all the things that could go wrong and creating unit tests for them. And that's where CoPilot was brilliant, as it came up with things that could go wrong that I had never thought of."

5 How and why do we interact with GenAI?

In the previous section, we described how GenAI is used for a variety of activities. In the studied companies GenAI was becoming an integrated part of their daily work, and most explained that they used GenAI daily, or "all the time".

Among the study participants we found two styles of interaction: *simple dialogue* and *advanced dialogue extended with prompt engineering*. The interaction style depended on the work context and types of problems to be solved. Table 3 contains an overview of the effects and drawbacks from interacting with GenAI.

5.1 Effects

By spending less time on manual and repetitive tasks, the improved productivity brought more enjoyment, motivation and fun to the work. The repetitive tasks were seen as menial, and not particularly mentally challenging. Further, as time was freed up, more time could be spent on creative and challenging tasks. Moreover, engaging with the GenAI itself was experienced as fun and increased the motivation to experiment with different applications of the new technology.

Interacting through dialogue with ChatGPT increased engagement. It was experienced as a more "natural" engagement than searching for answers to problems on Google. Having a dialogue with ChatGPT was also described as faster than concocting the necessary string of Google search keywords. Moreover, ChatGPT responds immediately with the, assumed, correct answer to the question, while googling often required additional steps, vetting the correct site on the search page, entering the particular page, and analysing the web page for the potential answer to the question. One informant was so conscious about speed that they deliberately chose GPT v3.5 over v4 in certain cases (at the time of

Table 3. Use of GenAI - Benefits and Challenges

Mode	Benefits	Challenges
Simple	<ul style="list-style-type: none"> – Asking questions is easier than searching – ChatGPT immediately provides an (almost) usable result – More efficient – More fun – More time to learn – Increased motivation and work satisfaction 	<ul style="list-style-type: none"> – Input cleaning – Lack of tool integration – Lack of information after 2021 – Output needs to be worked on – Culturally biased output
Advanced (Simple and Prompt engineering)	<ul style="list-style-type: none"> – Interactive learning – More precise answers – Ability to take on different roles – Pair-prompt engineering – ChatGPT as advisor - rubberducking 	<ul style="list-style-type: none"> – Limited context ability – Less pair-programming – Still requires other people when the complexity increases – Prompt engineering requires competence

our data collection v3.5 was faster than v4) , where the precision and quality of the v3.5 answer was assumed to be sufficient.

GenAI’s utility also extended beyond quick and precise answers, with informants reporting a freedom in interacting with an artificial tool rather than having to deal with the social considerations involved in asking a team member.

”Yeah, so you don’t need to be too polite either. You don’t have to have the correct phrasing or anything. You can just throw something out, I feel. Then you can get an answer, and if it’s not quite right, you can refine the question again.”

The threshold of asking ChatGPT was significantly lower than asking another person or in a Slack channel. This threshold for asking colleagues could potentially be high in a busy work environment, as one does not wish to interrupt an already-busy colleague. Further, you get feedback immediately, while it might take a while to get feedback on Slack.

5.2 Challenges

While there are many positive benefits of using GenAI, there are also challenges. The elusiveness of data confidentiality, data policy, and sensitivity meant that everyone was acutely conscious about which data to input into the chat interface. This made the work process somewhat awkward, requiring cleansing and

anonymization of the text being sent into ChatGPT. Developers in companies using open-source technology were less lenient in protecting code than those in companies with internal code repositories. Moreover, the general lack of tool integration meant that there was a substantial amount of copy-paste to move text and code between different windows. One developer using Copilot X reported that the integration in the IDE meant that the code could be autocompleted, and explained by ChatGPT in a seamless process, which reduced their work immensely: *"I think it would have been easier to adopt a GenAI tool if I had used something like Copilot. Because then it would have been, in a way, integrated into the workflow."*

With regards to the output, all informants noted that the content produced by GenAI, regardless of tool, seldom represented a final product and typically required further refinement to be applicable in a real-world context. The general attitude from the interviewees was that they expected the output to be wrong.

Regarding technological development, which is characterized by a rapid pace and a increasing number of available libraries and technology, the cut off date for ChatGPT's training data in September 2021 represented a significant drawback, where the error-rate was annoyingly high.

One architect creating project startup documents, experienced that ChatGPT was culturally biased towards how more hierarchical companies would perform activities in a project. This meant that ChatGPT had to be prompted with specific information regarding the methodology and project practices:

"You kind of have to trick it into the right context if it's (GenAI) going to be part of agile processes."

5.3 Prompt engineering

Several interviewees talked about how the quality of their prompt affected the quality of the response, and how the use of prompt engineering techniques like contextualizing the problem, using personas, etc. to guide and steer the dialogue with ChatGPT. Prompt engineering was applied to all the activities in table 2. One informant explained asking ChatGPT to create a description of the most critical code reviewer in the world. They then told ChatGPT to act like this description while reviewing the code in a pull request. Another more technical aspect was telling ChatGPT to act like an SQL database to test queries. The effect of using prompt engineering was seen as a matter of precision and quality, thus reducing the time spent on working on modifying the output. One explained the usage of prompt engineering as follows:

"It's like putting up fences on the bowling lane and then narrowing it down even more. It can almost only go one way, and that's a strike."

An important prompt engineering technique was assigning different roles to ChatGPT for the same question to get more than one perspective or answer on a problem. One explained,

"I want you to respond like a wealth manager," "I want you to respond like a friend," or "like a so-and-so..." And then you get different answers."

Using prompt engineering while writing code was described as feeling similar to programming with a partner. The flip side was that the developers mentioned that they were doing less pair programming as they were getting the wanted rubber ducking effect from using GenAI.

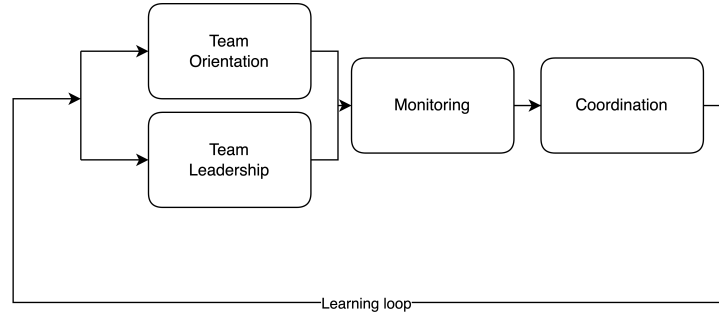
6 Discussion

One of the notable consequences of integrating GenAI tools into software development tasks is a visible shift in collaborative communication dynamics. Some of the informants appear to have a growing inclination to consult AI-driven solutions for issues and tasks they previously discussed with their human colleagues. This shift can have dramatic effects on the team’s ability to perform. According to Liu et al., [19], a team’s ability to perform is highly dependent on the knowledge sharing of the team. This implies that reducing knowledge sharing by replacing this with Generative should be observed. This opposes the findings by Brynjolfsson et al. [7], where knowledge dissemination between high-skilled and low-skilled workers in customer service. These findings point to a significant difference between work done by teams in software development and individual work in customer service. However, we find that individuals become more efficient and save time, which they spend on more rewarding tasks.

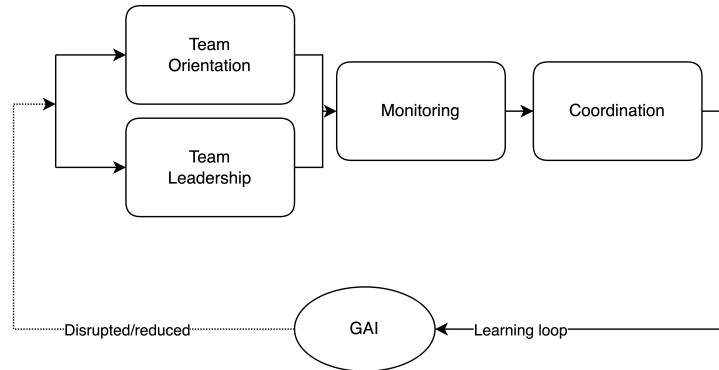
If team members reduce their interactions in favor of focusing on individual tasks, a phenomenon known as an isomorphic team structure may emerge. The advantages of this structure are that it is organizationally simple, allows many tasks to be completed in parallel, and can clearly define and understand task responsibilities. However, the effect of such a structure is that the developers focused on their own modules and often created their own plans and made their own decisions. In addition, problems are seen as personal, individual goals are more important than team goals, and team members become less aware of what others are doing and get less support and help from others [23]. In a good working team, learning is a continuous feedback, see Figure 1(a). By introducing GenAI; this loop will be disrupted, or reduced, Figure 1(b), thus reducing teamwork performance[23]. Additionally, this can also contribute to making persons less satisfied as helping others is a key factor for good workdays [22].

This model posits that the incorporation of GenAI in software development may disrupt the established learning loop. Such disruption will subsequently affect individual and team performance in software development. While it is anticipated that GenAI might enhance individual performance by streamlining tasks, there is a concurrent risk of diminishing overall team performance.

However, everything is not dark; as multiple informants noted, they had an increasing amount of knowledge sharing on how to use GenAI in their work and context. Notably, the practice of ”pair prompt engineering” has emerged, akin to the concept of pair programming. This approach facilitates knowledge sharing [34] both on-site and when working remotely. This can thus involve a shift in how the programmers program, creating yet another abstraction layer for code production.



(a) Regular learning loop



(b) Disrupted learning loop

Fig. 1. Regular and disrupted learning loops.

7 Concluding remarks

In essence, GenAI serves a dual purpose: making everyday tasks more efficient and reigniting creative thinking for leaders and developers. By automating the production of routine code snippets and related tasks, these tools enable programmers to focus on higher-level conceptualization and innovation, resulting in enhanced productivity and code quality. This is similar to findings by Meyer et al. [22], where good workdays are understood as days where they feel productive and are able to work focused. In addition, similar to Brynjolfsson et al. where there is knowledge dissemination through the GenAI [7], we observe that there are data scientists using GenAI for coding purposes and frontend developers getting assistance in back-end development. Both programmers and leaders acknowledged the potential of generative AI in freeing up valuable time and cognitive resources that could be better allocated to more creative and complex problem-solving tasks.

This is reported to enhance both efficiency and enjoyment. By automating the production of routine code snippets and documentation, these GenAI enabled programmers to focus on higher-level conceptualization and other more

complicated tasks, resulting in higher reported productivity. Most informants reported that AI reduced the amount of time developers spent on projects.

Acknowledgments

This work was supported by the Research Council of Norway grants, 321477, and 309344, and the companies Knowit, and Iterate through the research projects Transformat and 10XTeams.

References

1. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: Can language models be too big? In: Proceedings of the 2021 ACM conference on fairness, accountability, and transparency. pp. 610–623 (2021)
2. Berntzen, M., Stray, V., Moe, N.B., Hoda, R.: Responding to change over time: A longitudinal case study on changes in coordination mechanisms in large-scale agile. *Empirical Software Engineering* **28**(5), 114 (2023). <https://doi.org/10.1007/s10664-023-10349-0>
3. Biernacki, P., Waldorf, D.: Snowball sampling: Problems and techniques of chain referral sampling. *Sociological methods & research* **10**(2), 141–163 (1981), publisher: Sage Publications Sage CA: Los Angeles, CA
4. Bird, C., Ford, D., Zimmermann, T., Forsgren, N., Kalliamvakou, E., Lowdermilk, T., Gazit, I.: Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. *Queue* **20**(6), 35–57 (Dec 2022). <https://doi.org/10.1145/3582083>, <https://dl.acm.org/doi/10.1145/3582083>
5. Bosch, J., Olsson, H.H.: Digital for real: A multicase study on the digital transformation of companies in the embedded systems domain. *Journal of Software: Evolution and Process* **33**(5) (2021). <https://doi.org/10.1002/smr.2333>
6. Bruch, M., Monperrus, M., Mezini, M.: Learning from examples to improve code completion systems. In: Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering. pp. 213–222 (2009)
7. Brynjolfsson, E., Li, D., Raymond, L.R.: Generative AI at work. Tech. rep., National Bureau of Economic Research (2023)
8. Chatterjee, P., Kong, M., Pollock, L.: Finding help with programming errors: An exploratory study of novice software engineers’ focus in stack overflow posts. *Journal of Systems and Software* **159**, 110454 (2020), publisher: Elsevier
9. Corbin, J., Strauss, A.: Basics of qualitative research: Techniques and procedures for developing grounded theory. Sage publications (2014)
10. Dell’Acqua, F., McFowland, E., Mollick, E.R., Lifshitz-Assaf, H., Kellogg, K., Rajendran, S., Kray, L., Candelon, F., Lakhani, K.R.: Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality (Sep 2023). <https://doi.org/10.2139/ssrn.4573321>, <https://papers.ssrn.com/abstract=4573321>
11. Dickinson, T.L., McIntyre, R.M.: A conceptual framework for teamwork measurement. In: Team performance assessment and measurement, pp. 31–56. Psychology Press (1997)

12. Dingsøy, T., Bjørnson, F.O., Shull, F.: What Do We Know about Knowledge Management? Practical Implications for Software Engineering. *IEEE Software* **26**(3), 100–103 (May 2009). <https://doi.org/10.1109/MS.2009.82>, conference Name: IEEE Software
13. Fitzgerald, B., Stol, K.J.: Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* (2015). <https://doi.org/http://dx.doi.org/10.1016/j.jss.2015.06.063>
14. Florea, R., Stray, V.: A global view on the hard skills and testing tools in software testing. In: 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE). pp. 143–151. IEEE (2019). <https://doi.org/10.1109/ICGSE.2019.00035>
15. Forsgren, N., Storey, M.A., Maddila, C., Zimmermann, T., Houck, B., Butler, J.: The SPACE of Developer Productivity: There’s more to it than you think. *Queue* **19**(1), 20–48 (2021)
16. Hindle, A., Barr, E.T., Gabel, M., Su, Z., Devanbu, P.: On the naturalness of software. *Communications of the ACM* **59**(5), 122–131 (2016), publisher: ACM New York, NY, USA
17. Kersten, M., Murphy, G.C.: Using task context to improve programmer productivity. In: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering. pp. 1–11 (2006)
18. Li, A., Endres, M., Weimer, W.: Debugging with stack overflow: web search behavior in novice and expert programmers. In: Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training. pp. 69–81. ICSE-SEET ’22, Association for Computing Machinery, New York, NY, USA (Oct 2022). <https://doi.org/10.1145/3510456.3514147>, <https://dl.acm.org/doi/10.1145/3510456.3514147>
19. Liu, M.L., Hsieh, M.W., Hsiao, C., Lin, C.P., Yang, C.: Modeling knowledge sharing and team performance in technology industry: the main and moderating effects of happiness. *Review of Managerial Science* **14**(3), 587–610 (Jun 2020). <https://doi.org/10.1007/s11846-018-0301-4>, <https://doi.org/10.1007/s11846-018-0301-4>
20. Melegati, J., Guerra, E.: Dante: a taxonomy for the automation degree of software engineering tasks. *arXiv* (2023)
21. Mens, T., Cataldo, M., Damian, D.: The Social Developer: The Future of Software Development [Guest Editors’ Introduction]. *IEEE Software* **36**(1), 11–14 (2019), publisher: IEEE
22. Meyer, A.N., Barr, E.T., Bird, C., Zimmermann, T.: Today was a good day: The daily life of software developers. *IEEE Transactions on Software Engineering* **47**(5), 863–880 (2019), publisher: IEEE
23. Moe, N.B., Dingsøy, T., Dybå, T.: A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology* **52**(5), 480–491 (2010), publisher: Elsevier
24. Moe, N.B., Stray, V., Smite, D., Mikalsen, M.: Attractive Workplaces: What are Engineers Looking For? *IEEE Software* (2023), publisher: IEEE
25. Muller, M., Ross, S., Houde, S., Agarwal, M., Martinez, F., Richards, J., Talamadupula, K., Weisz, J.D.: Drinking Chai with Your (AI) Programming Partner: A Design Fiction about Generative AI for Software Engineering. In: HAI-GEN Workshop at IUI 2022: 3rd Workshop on Human-AI Co-Creation with Generative Models (2022), <https://hai-gen.github.io/2022/papers/paper-HAIGEN-MullerMichael.pdf>

26. Murphy, G., Kersten, M., Findlater, L.: How are Java software developers using the Eclipse IDE? *IEEE Software* **23**(4), 76–83 (Jul 2006). <https://doi.org/10.1109/MS.2006.105>, conference Name: IEEE Software
27. Ravn, J.E., Moe, N.B., Stray, V., Seim, E.A.: Team autonomy and digital transformation: Disruptions and adjustments in a well-established organizational principle. *AI & SOCIETY* **37**(2), 701–710 (2022)
28. Ross, S.I., Martinez, F., Houde, S., Muller, M., Weisz, J.D.: The programmer’s assistant: Conversational interaction with a large language model for software development. *Proceedings of the 28th International Conference on Intelligent User Interfaces* p. 491–514 (2023). <https://doi.org/10.1145/3581641.3584037>
29. Saldaña, J.: *The coding manual for qualitative researchers*. SAGE, Los Angeles, 2nd ed edn. (2013), oCLC: ocn796279115
30. Stray, V., Gundelsby, J.H., Ulfsnes, R., Brede Moe, N.: How agile teams make Objectives and Key Results (OKRs) work. In: *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering*. pp. 104–109 (2022)
31. Sun, J., Liao, Q.V., Muller, M., Agarwal, M., Houde, S., Talamadupula, K., Weisz, J.D.: Investigating explainability of generative AI for code through scenario-based design. In: *27th International Conference on Intelligent User Interfaces*. pp. 212–228 (2022)
32. Svyatkovskiy, A., Zhao, Y., Fu, S., Sundaresan, N.: Pythia: Ai-assisted code completion system. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. pp. 2727–2735 (2019)
33. Talamadupula, K.: Applied AI matters: AI4Code: applying artificial intelligence to source code. *AI Matters* **7**(1), 18–20 (Mar 2021). <https://doi.org/10.1145/3465074.3465080>, <https://dl.acm.org/doi/10.1145/3465074.3465080>
34. Tkalic, A., Moe, N.B., Andersen, N.H., Stray, V., Barbala, A.M.: Pair Programming Practiced in Hybrid Work. In: *Proceedings of the 17th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Association for Computing Machinery, New York, NY, United States (2023)
35. Tkalic, A., Ulfsnes, R., Moe, N.B.: Toward an agile product management: What do product managers do in agile companies? In: *International Conference on Agile Software Development*. pp. 168–184. Springer (2022)
36. Ulfsnes, R., Mikalsen, M., Sporse, T., Hatling, M.: Technology for knowledge work: A relational perspective. *ECIS 2023 Research-in-Progress Papers* (May 2023), https://aisel.aisnet.org/ecis2023_rip/48
37. White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C.: Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. *arXiv* (2023). <https://doi.org/10.48550/arxiv.2303.07839>
38. Yin, R.K.: *Case study research and applications: Design and methods*. Los Angeles, UK: Sage (2018)