Exploring potential implications of intelligent tools for human aspects of software engineering

Jorge Melegati
jorge.melegati@unibz.it
Free University of Bozen-Bolzano
Bolzano, Italy

Nicolas Nascimento
nicolas.nascimento@pucrs.br
PUCRS
Porto Alegre, Brazil

Rafael Chanin
rafael.chanin@pucrs.br
PUCRS
Porto Alegre, Brazil

Afonso Sales
afonso.sales@pucrs.br
PUCRS
Porto Alegre, Brazil

Igor Wiese
igor@utfpr.edu.br
Federal University of Technology - Paraná (UTFPR)
Campo Mourão, Brazil

ABSTRACT

Background. The emergence of tools based on artificial intelligence (AI) to support software development suggests an overhaul on how developers program and interact among themselves. This disruption might bring challenges regarding human and social aspects of the software development process. *Objective.* This paper is a first exploration of the consequences of AI-based tools for software development teams and their members. *Method.* We conducted a social science fiction exercise, a sort of thought experiment, narrating two fictional stories about a futuristic software company employing AI-based tools. Then, we evaluated the plausibility of one of the scenarios through a qualitative experiment with 38 students to observe their perception regarding the use of AI-based tools. *Results.* The stories suggest potential challenges related to the adoption of these tools: a change on how developers perceive themselves, a clash between quantitative and qualitative worker contribution assessment, and the training of future developers to handle the imminent changes on their profession. In the qualitative experiment, we collected evidence supporting negative feelings, such as lack of trust and control and fear of being replaced. We also identified other attitudes and perceptions of developers, such as positive feelings towards AI-based tools. *Conclusion.* We identified several aspects that might influence the adoption of AI-based tools and their implications for individuals involved. They should be further investigated and represent a challenge for the research on human aspects of software engineering. We also demonstrated the use of social science fiction to explore novel research problems.

CCS CONCEPTS

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

CHASE '24, April 14–15, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0533-5/24/04.
https://doi.org/10.1145/3641822.3641877

KEYWORDS

AI for SE, social science fiction, human aspects of software development, qualitative experiment

ACM Reference Format:
Jorge Melegati, Nicolas Nascimento, Rafael Chanin, Afonso Sales, and Igor Wiese. 2024. Exploring potential implications of intelligent tools for human aspects of software engineering. In *2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE '24), April 14–15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3641822.3641877

1 INTRODUCTION

Although software development emerged less than a century ago, it has experienced several changes in how its activities are performed from technical and organizational perspectives. In the beginning, around 1960, individuals developed small programs based on an elementary understanding of control flows using mnemonics [37]. After 60 years, the process evolved through developing higher-level languages, abstract architectures, and larger systems but also with expanding development teams [37, 38], sometimes spread around the globe. In this evolution process, several tools or approaches changed the way developers as human beings interact among themselves and with technology. For instance, Integrated Development Environments (IDE) continuously add new features to automatize repetitive tasks, such as refactoring [28], along with agile methods [1] and global development teams [15] that brought several changes and new challenges for how developers collaborate.

However, these changes seem minor compared to the impeding disruptive arrival of a new set of tools enabled by artificial intelligence (AI). An example is the launch of GitHub Copilot[1], a tool that synthesizes code based on machine learning (ML) algorithms trained with code from repositories hosted in the GitHub platform. Another has been the use of OpenAI's ChatGPT[2] to support software engineering (SE) tasks. These tools are probably the first of a series of productivity enhancers enabled by AI and ML. This forecast does not surprise those who observe the prominence of this topic in premier venues for SE research.

Although research in SE has focused on how to use AI to enhance development activities [8, 9, 46], it has neglected the impact of this

[1] https://copilot.github.com/
[2] https://chat.openai.com/

usage on individuals and social groups, i.e., developers, teams, and organizations. In this paper, to demonstrate potential problems, we conducted a social science fiction exercise, a sort of thought experiment, to explore the potential implications of AI-based tools for individuals involved in the software development process. Based on these stories, we identified a set of propositions describing potential issues of AI-based tools to developers and teams. Then, we evaluated two of the propositions, namely the fear of being replaced by tools and the influence of these tools on how developers perceive themselves, through a qualitative experiment. It consisted of proposing a programming problem for a group of students in which AI-based tools usage was controlled. We applied questionnaires to the participants and qualitatively analyzed the results. Based on that, we discussed the potential implications of AI-based tools for human aspects of SE and also the use of the social science fiction method for SE research.

Our contributions are three-fold: 1) a list of propositions regarding possible impacts of AI-based tools on human and social aspects of SE; 2) some pieces of evidence to support two propositions regarding how developers perceive themselves and their feeling of dependence towards the tools; and 3) an approach to employ the social science fiction for SE research by using it to generate propositions to be further evaluated with empirical studies.

The remainder of this paper is organized as follows. Section 2 presents the background and related work. In Section 3, we present the social science fiction study, describing the method, the stories produced, and the propositions reached. Section 4 presents the qualitative experiment we conducted and Section 5 describes the results obtained. In Section 6, we discuss our results in relation to the existing literature. Finally, Section 7 concludes the paper.

2 BACKGROUND AND RELATED WORK

In her seminal paper about an engineering discipline for software [37] published in 1990, and a follow-on paper [38] published in 2009, Mary Shaw describes how software engineering and the research problems associated with it evolved since the creation of the first computers. From the beginning until around 1965, programming consisted of small programs built in any way possible with simple structures. Around 1970, the focus shifted to programming in the small and the first high-level languages emerged. Around 1980, the programming-in-the-large began with complex and continuously executed systems. Around 1990, abstract architectures guiding software design emerged. From an organizational perspective, the complexity also changed from individual efforts to team efforts to distributed teams. The emergence of the World Wide Web led, around 2000, to an explosion of not only consumers but also producers of applications. In the 2009 paper, she speculated that challenges would emerge on the boundaries of very complex systems and users. In her own words: "Problems facing software engineers are increasingly situated in complex social contexts, and delineating the problems' boundaries is increasingly difficult."

Shaw's speculation was right, and the human and social consequences of software systems are increasingly noticed. In a 2020 *Communications of ACM* paper [4], Connolly presents his opinion that computer science should be closer to social sciences. He argues that researchers could cope better with the complexity of subjects computer science is facing using the "methodological and theoretical pluralism" from the social sciences rather than relying on a "single methodological approach for making and evaluating knowledge claims" from natural and engineering sciences. This call is echoed by Mendez-Fernandez et al. [23] that stress the increased need for interdisciplinary research in software engineering, especially "for the investigation of social, cultural, and human-centric aspects of software engineering." However, in SE research, although the use of other paradigms, such as interpretivism-constructivism, has started, the majority of studies still rely on a pragmatic paradigm heavily influenced by post-positivism [22].

Recently, the rise of AI-enabled software led researchers to focus on the impacts these systems have on the user, specifically, AI ethics, i.e., the ethical considerations about the AI software for users and the potential responsibilities developers have [18, 41, 44]. However, in industry, this concern is still in its infancy, and AI has been considered just another feature [41]. Bryson and Winfield [2] describe several initiatives from the industry, professional associations, and government towards ethical standards for intelligent systems, such as the British Standard 8611:2016 "Robots and Robotic Devices: Guide to the Ethical Design and Application of Robots and Robotic Systems." The authors, in special, are working on a transparency standard as part of a larger initiative from IEEE. In this draft work, they identify five categories of stakeholders: users, safety certification agencies, accident investigators, lawyers or expert witnesses, and broader society. An autonomous system should be transparent for each class of stakeholder in a different way and for different reasons.

Not only has software implications for humans, but also individual and social aspects influence the software development process. In this sense, several studies have investigated the influence of human aspects on the productivity of software development processes. In a survey with 317 responses, Graziotin et al. [11] identified 42 consequences of developers' unhappiness and 32 of happiness. The most common consequences of unhappiness were external, including low productivity and code quality, but also consequences to the developer's own being, including low cognitive performance and mental unease and disorder. Another important aspect of software development teams is turnover intention. When a member leaves the team, there might be tacit knowledge that is lost with that person. Besides that, the team will need to spend time and resources to recruit and train a substitute. In this regard, job satisfaction is associated with a lower turnover intention [36]. These results explain why research in human aspects of software engineering expanded, focusing not only on the impact of happiness in productivity but also the triggers for emotions during software development and their impact on the developers' well-being [27]. To the best of our knowledge, there are no studies focused on the implications of the use of AI-enabled tools for software developers regarding human and social aspects. So far, research has focused on technical challenges, e.g., [16] and [20].

3 A SOCIAL SCIENCE FICTION EXERCISE

As a first step to tackle this research problem, we performed a social science fiction exercise. This method is a thought experiment in the

form of fictional stories that explore how technological advancements might impact social life and social order [10]. It is associated with the idea that the future is not a natural fact but a design decision, i.e., it is built. As Montfort [26] writes: "The future is not something to be predicted, but to be made".

Although we have not found previous uses of the method in SE research, there are examples in related fields. For technology-enhanced learning, Selwyn et al. [35] performed a social science exercise to speculate how using digital tools to support education will affect schools in the next decade. Based on five vignettes about characters from an Australian high school, including students, their parents, and teachers, they identified three potential consequences of this phenomenon: a reconfiguration of time and space of school, i.e., students will have to engage with the school on a continuous space regardless of time and space; the increasingly intertwinement between school space with code, changing how social process occur including power relations, the increasing "platformization of the school," and the increased dependency of the school to data digitally generated.

3.1 Procedure

To increase the study's reliability, we followed a defined process to create the stories. Another objective was to avoid creating fantastic, unfeasible narratives. It is essential to stress the *social science* aspect and to remind that we are not doing *science fiction* literature. First, we reviewed recent studies on AI for SE published in top venues and tools available for practitioners, specifically automatic code generation and bug prediction. Besides GitHub Copilot, regarding automatic code generation, we can mention IntelliCode Compose [39], a tool "capable of predicting sequences of code tokens of arbitrary types, generating up to entire lines of syntactically correct code." For bug prediction, the results obtained by Di Nucci et al. [7] caught our attention: their model using developers' structural scattering, a metric to show how scattered the code modifications introduced by a single developer, was superior to other bug prediction models. The authors stress that developer-related factors could still be further explored for bug prediction.

Inspired by these models and tools, we extrapolated the potentialities of these tools and drafted two stories. Then, we presented the two drafts in a writers' workshop, where a group of software engineering and information systems researchers discussed ideas and initial versions of research papers. Our goal was to obtain feedback about the stories and verify if they represented reasonable scenarios for the future. The group consisted of six researchers, including one of the authors, mainly at the initial stages of their research career, including PhD students, postdocs, and an assistant professor. The feedback was good, and the participants only suggested small story changes.

3.2 Stories

As a background for the stories, we will use a fictional company called Awesome. It offers Software-as-a-Service (SaaS) productivity tools to creative graphical workers, such as designers. An internal development team is responsible for implementing the software. It consists of software programmers but also of user interface designers, user experience experts, quality assurance specialists, testers, and managers. Besides this engineering team, the company has other teams, such as marketing, sales, operations, and administrative staff.

With Awesome, we aim to represent a typical software-intensive company in which a number of teams develop software for external users. We expect that, in the near future, most of these companies will employ AI-enabled tools to support software development to improve productivity and improve code quality. To represent these tools, we propose imaginary but feasible near-future solutions available to developers and managers of Awesome. The first is Navigator, a commercial IDE implementing cutting-edge machine-learning algorithms to enable coding assistance, such as advanced autocompletion and bug prediction. The second is the Quality Management System (QMS), an automatic tool to assess the quality of the code produced continuously. Below, we describe fictional situations of how these tools could affect the lives of collaborators of the company.

3.2.1 Do I really know how to code? Miriam was really happy. Even after a year working at Awesome, she was still jubilant. It was the job she dreamed of during her Software Engineering bachelor studies. Of course, the job is much easier than she thought. She often writes a method name and the Navigator tool creates a draft on which she needs to do some small tweaks. Once, Miriam explained her job to her grandfather. He replied that it had reminded him of when the new robots arrived in the car factory where he used to work until retiring. Since this fact, Miriam started wondering if, in the future, she would be substituted by a robot like many of her grandfather's colleagues were in the factory. But she was not thinking about her future that day; she was enjoying being a software engineer at Awesome. When she wrote the first method signature, no suggestions appeared. *'What happened?',* she thought. She looked around and saw several interrogation faces among her colleagues. In a matter of minutes, the situation was clear: Navigator was offline. The provider of the autocompletion tool had been hacked, and it would not work for the remainder of the day. The situation represented a problem for Miriam who was handling a critical bug preventing a fraction of users from accessing a tool provided by Awesome. At first, Miriam thought it would be fine: before joining Awesome, she had never used Navigator and managed to code several projects during her bachelor studies. *"It is going to be fun,"* Miriam thought. But her mind changed quite soon as she had to code relying only on her mind and simple autocompletion tools that did not rely on Navigator. *"How do I do this structure in this language? And what is this call to develop this task using this API?"* At the end of the day, Miriam had managed to finish the task, but she was not sure if she had done the best job. Without Navigator, it was not possible to assess it. On her way home, several doubts came to her mind: *"Am I really a software developer? What am I? Just a tool operator?"*

3.2.2 The algorithm is firing you. That day seemed an ordinary one for Kyle. He was going to the same address he had gone to for five years since he joined the startup as the second developer in the Awesome team. The address is the same but not the place. When he started, the team comprised eight people working in a 50-square-meter office. Now, the company has 150 collaborators distributed in three floors. He was the first hired developer in the

company. Before him, there was only Martin, one of the founders with a technical background, who used the title of CTO.

When Kyle enters the engineering team's open space, he passes by Martin's glass office. The CTO calls him for a talk. The developer thought it would be another conversation about the new architecture they are building to support the company's expansion. However, this idea changes when Kyle enters the room and sees Martin's face. He had seen a similar face once when he was dismissed from his first job. He imagines that the subject of the conversation would be similar.

The fear of being removed from that place, the company he witnessed to grow and conquer the dreams they had in that small room, had installed in his mind. The feeling increases as Martin describes Kyle's importance in the company's building. *'Martin, are you firing me?' 'I'm sorry, Kyle. I don't have a choice. QMS says that the code you make has a probability, on average, of 30% of being buggy. You can see that on your personal dashboard, no?'* One of QMS' features is to predict, based on an AI-enabled algorithm, the probability that a piece of code is buggy or not. *'You know, since the IPO, QMS reports are sent to the shareholders. If we keep you, there will be a flag on the report... saying that we decided to keep a developer with a tendency to create buggy code. And, you know... it is really hard to talk to these people... I'm really sorry...'* Kyle tries to argue *'But I know every detail of this system. I worked in architecture from the beginning... Because of that, I often handle more tricky problems... You know... this could explain why...'* But the decision has been taken and today is the last day of Kyle at Awesome.

3.3 Potential issues

The stories presented above portray potential issues for developers of AI-enabled tools.

First, developers will probably experience a disruption in their importance to the software development process. Rather than being those responsible for generating code, they might only supervise automatic code creation. This development is similar to what happened in manufacturing where machines replaced manual workers. Work has been extensively studied as a significant element of self-existence. From psychological and sociological aspects, philosophers and researchers, such as Marx, Durkheim, and Weber [3], described how society is based on the productive activities of human beings. As we have seen in both stories, developers might face existential questions when they perceive being replaced by intelligent tools. The implications of this substitution for knowledge workers, of which software engineers are considered the cutting edge [17], is yet to be seen. Based on this discussion, we can reach the following propositions:

> **Proposition 1**: *Developers might feel replaced by AI-based tools.*

> **Proposition 2**: *AI-based tools might influence the self-perception of developers.*

Then, as presented in Kyle's story, decision-making will be increasingly delegated to programmatic solutions to avoid biases and errors from human managers. However, automatic solutions might not consider non-tangible, qualitative aspects such as tacit knowledge and teamwork. Tacit knowledge is a key aspect in software development, associated with more effective teams [33]. This issue is related to the ever-increasing surveillance in the societies of the future [30]. We posit this issue as the following proposition:

> **Proposition 3**: *The delegation of decision-making to AI-based tools might depersonalize decisions.*

Finally, a challenge that the emergence of AI-enabled tools poses to technology educators is how to prepare students to handle the impending changes they will face during their work. A current student will probably work for more than 30 years in the field. The first story shows the story of Miriam who had just completed her training and is already lost on how her professional life will be. The following proposition represents this issue:

> **Proposition 4**: *Education and training of software developers need to be adapted to the use of AI-based tools for SE.*

4 A QUALITATIVE EXPERIMENT

By performing the social science fiction exercise, we reached several potential individual and social issues regarding AI-based tools. In the next step, we aim to collect empirical data to support or refute these propositions. Unfortunately, performing studies to evaluate all the propositions identified is not feasible in a single study, not even a single paper. Given that AI-based tools are still being adopted in software development teams and companies, evaluating the implications at the team level would not be currently feasible. Therefore, in this next step, we will focus on the first two propositions raised in the social science fiction exercise, namely a potential feeling of replacement felt by developers towards AI-based tools and the influence of these towards developers' self-perception. Through this study, we also aim to demonstrate the value of this method for the research on human aspects of SE.

Therefore, we decided to explore the feelings felt by developers towards AI-based tools, through a qualitative experiment, an integrated approach consisting of investigating the process of meaning construction in an experimental setup [32]. Qualitative experiments employ qualitative strategies, such as focus groups and interviews, on subjects exposed to a randomized stimulus, as in a typical quantitative strategy [32]. By performing this experiment, we aim to verify the existence of evidence supporting or not the two propositions.

A qualitative experiment is composed of four phases [32]. In the first phase, the pre-test, participants answer a questionnaire to be profiled. In the second phase, participants are exposed to a stimulus, and their reactions are collected in the third. Finally, in an optional fourth phase, the participants' reactions to the whole process are collected. Below, we describe the qualitative experiment we conducted including the participants and the context in which they are inserted. A summary of the experiment execution is presented in Figure 1.

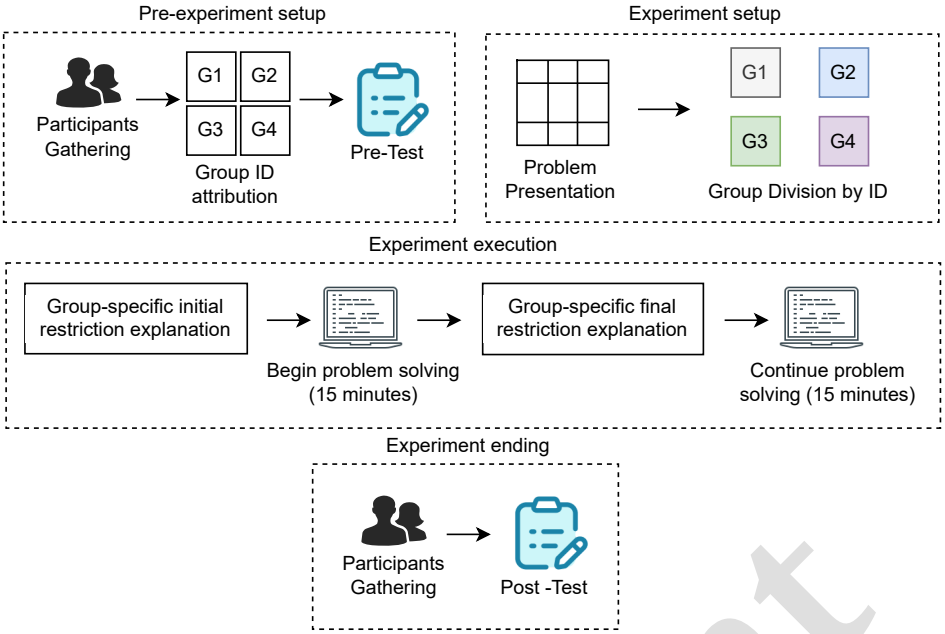

Figure 1: Experiment execution steps.

4.1 Participants

The experiment was conducted in an R&D environment in a Brazilian university that teaches Apple mobile application development. In the environment, students are introduced to the development ecosystem of Apple platforms and obtain experience by working together in agile teams. Students can focus on software development or interface design during the program but are exposed to both areas during their training. These students are supported by tech, design, and business mentors who have solid experience in the software industry. We will refer to these mentors as the leadership team in this paper. The teams work on developing apps that support users to overcome real-world problems. The environment encourages team members to cooperate in all software development stages from requirements elicitation to deployment and maintenance steps. All teams use Scrum [34] to manage their work.

4.2 Pre-test

In the pre-test phase, we aimed to collect the participants' perceptions regarding the role of a software developer and AI-based tools for SE. To achieve this, we developed a questionnaire consisting of open-ended questions which is available in Appendix A.

4.3 Stimulus

We designed the experiment with 4 different experimental conditions (G1, G2, G3, and G4), depending on how participants should use AI-based tools to solve a non-trivial programming problem. Participants in G1 would never be allowed to use AI tools (control group). Participants in G2 would begin the problem by not using any tools but would start using generative AI-based tools in the second half of the experiment. Participants in G3 would be the opposite: in the first half, they could use any tool, and then, in the second half of the experiment, they would be instructed to stop using any tools. Finally, participants in G4 would always have to use generative AI-based tools.

The programming problem of the experiment had to be non-trivial to make it difficult for the generative AI-based tools to provide the answer on the first attempt. Participants would have to provide many prompts to eventually get the final answer from the tool. In this sense, our strategy entailed selecting a well-known math problem "masked" as a set of statements the participants should follow. The chosen problem was the power method. The power method is a numerical algorithm used primarily to estimate the largest eigenvalue of a matrix and its corresponding eigenvector. It is particularly useful when dealing with large matrices where other methods might be computationally impractical. The algorithm is iterative, starting with an initial guess for the eigenvector and repeatedly multiplying it by the matrix until convergence.

This problem was presented to the participants as follows:

- Vector of "n" positions that multiplies a square matrix (n x n);
- Initial vector with its first position equal to 1.0, i.e., v[0] = 1.0. The remaining positions are equal to 0.0;
- Values in the positions of the matrix should be between 0.0 and 1.0 (not inclusive);
- The sum of each row of the matrix must be equal to 1.0;
- Iterative process, with stopping condition: values in all vector positions must respect a tolerance of 6 decimal places.

As an example, we presented the vector $\begin{bmatrix}1.0 & 0.0 & 0.0 & 0.0\end{bmatrix}$ and the following matrix:

$$\begin{bmatrix} 0.25 & 0.20 & 0.10 & 0.45 \\ 0.10 & 0.45 & 0.35 & 0.10 \\ 0.07 & 0.90 & 0.02 & 0.01 \\ 0.80 & 0.05 & 0.08 & 0.07 \end{bmatrix}$$

Then, we went step-by-step explaining the expected outcome until we got to the final result (the expected vector respecting the given condition). Participants were allowed to use the same vector and matrix to check whether they were going correctly.

4.4 Post-test

In this step, in which we combined the third and fourth phases, participants answered another questionnaire with open-ended questions that were different depending whether they had used an AI-based tool or not. Participants in the control group (G1) assessed the task's difficulty and if it had changed their perception of the role of a software developer. The other participants had to answer questions regarding their impressions and opinions of AI-based tools if they depended on the tools, and if the experiment changed their perception of the software developer role. These instruments were also validated in the pilot and are available in Appendix A.

4.5 Pilot

In order to test our experiment, we ran a pilot with four members of the leadership team who were not involved in the research. Hence, they did not have any information about this process. Each member represented one of the groups (G1, G2, G3 and G4).

We began the pilot by assigning each one an ID (from 1 to 4), and then we ran the pre-test. We found out that on questions 4 and 5 (see Appendix A), it was important to add clear examples of what we were talking about. Therefore, we clarified the questions by adding the GitHub Copilot example.

After that, we gave each one a piece of paper on which they could read the instructions about the experiment (in this case, whether or not they could use AI-based tools). No questions were asked about these instructions. Hence, we moved to the next step which was the explanation of the problem (as explained in Section 4.3). The participants understood the problem well.

It is important to note that we estimated, based on our own experience, that the problem could be solved in 20 minutes. Therefore, we planned to run the experiment for 10 minutes and then give participants the new instructions for the remaining 10 minutes.

At this moment, we told participants they would have 20 minutes to solve the problem, and we started the clock. When we reached 10 minutes, we gave them their new instruction and let them work for the remaining 10 minutes. None of the participants have finished on time, but they were pretty close.

To finish the pilot, we asked them to respond to the post-test questionnaire. We thanked the participants who participated in the activity and ended the process. Now, we were ready to improve our experiment.

Our first modification from the pilot was the simplification of some questions, as the pilot participants reported that the questions were not clear and there were typos in the questions. The second and final modification was regarding the time. Given that pilot participants reported that they could not finish on time (and knowing they had more experience than the actual study participants), we decided to increase the time of the experiment to 30 minutes. With this modification, we proceed to execute the experiment.

4.6 Execution details

For our study, we invited 38 participants who were part of this R&D environment and were focusing on software development. The participation was voluntary and participants could leave the experiment at any moment. For the experiment, we followed the steps below:

(1) The researchers gathered the participants in an auditorium;
(2) The researchers attributed a random identifier (G1, G2, G3 or G4) at random to each participant;
(3) Participants fulfilled the pre-test questionnaire;
(4) The researchers presented the programming problem that they should solve. During this step, participants had the chance to ask any questions related to the programming problem;
(5) Participants were separated into four groups based on their identifier;
(6) Participants of each group were then oriented on the specific restrictions of their groups. G1 and G2 were initially oriented not to use any generative AI solution to solve the programming problem and G3 and G4 were initially oriented to use any generative AI solution to achieve this. This step lasted 15 minutes.
(7) The experiment was paused and new instructions were given to the participants. G1 and G4 were not given any new instructions (in other words, they were told that the original instructions remained valid). G2 was given a new instruction that from that moment forward, they were oriented to use any generative AI solution to solve the programming problem and G3 was given a new instruction that from that moment forward, they were oriented not to use any generative AI solution to solve the programming problem. This step lasted 15 minutes.
(8) The researchers gathered the participants in the auditorium again;
(9) Participants filled out the post-test questionnaire.

4.7 Data analysis

The participants answered the questionnaires on an online platform. The answers were collected and saved in a spreadsheet. The first step of the qualitative analysis consisted of open coding. Since we focused on openly identifying themes rather than labeling the data using predefined codes, we did not calculate the Inter-Rater Reliability (IRR) agreement. According to McDonald et al. [21], this open coding approach does not need the calculation of IRR.

Initially, one of the authors conducted this process alone, and afterward, a second author reviewed the codes. Finally, all the authors discussed potential issues and fixed the codes accordingly. In this process, answers not related to the goal of the experiment were discarded. The final codes were then classified as being positive towards the tools, negative, or indifferent. Finally, we analyzed how the answers were classified depending on the stimulus the participants received. The following section presents our results.

4.8 Replication package

To allow the replication of our study, we provide a replication package[3] consisting of all the answers freely translated to English[4], redacting sensitive information. We also present how we coded each answer.

5 RESULTS

In this section, we present the results of our analysis. First, we present the codes we identified related to developers' perceptions of the tool, and then, we compare the answers of the different groups.

5.1 Developers' perceptions about the tool

To allow the comparison of the different groups of participants, we divided the developers' perception into two high-levels categories: negative and positive perceptions. Below, we describe the categories present in each one of these high-level categories.

5.1.1 *Negative perceptions.*

Negative feelings. This category groups codes related to negative feelings expressed by the participants related to the use of the tool. First of all, some participants mentioned aspects regarding a possible dependence of the tool to perform their activities. Still in the pre-test questionnaire, a participant already described what could be a **risk of getting dependent**: *"I see several peer students using these tools daily for everything, I perceive this creating a certain dependency on these technologies and reducing the capacity of developing software by their own."* When asked about the issue, many participants described **dependence of the tool**: *"I believe I realized how much I was dependent [when trying] to debug something."* Besides concerns related to a possible dependency to the tool, participants expressed other negative feelings towards the tool. First, a participant mentioned the **lack of control**: *"[Once we were allowed to use the tool,] I tried to see if it could answer the question of the challenge using my code as a basis. [...] it delivered to me another implementation code [of which] I lost control, I did not understand what many things were doing and, consequently, [the solution] was not working."* Another common feeling expressed by the participants was the **lack of trust** on the tool. In this regard, they have mentioned several times that they struggle to rely on the solutions given by the tool since they know that the tool might make mistakes. A participant wrote: *"I feel that it is hard to know [until when] one can trust [ChatGPT]"*. Finally, a participant was **scared by the tool capacity**. When answering the question about the feelings after starting using the tool, a respondent answered: *"Stunned, because I had already tried for several minutes to understand the problem, but when I gave the same instructions to the tool, it gave me the solution and the detailed explanation in seconds."* Other participants expressed negative feelings towards their own selves motivated by the use of the tool. A participant expressed **guilty about not doing the task**: *"I felt a bit guilty for not writing the code and thinking of the logic, but [the tool] made the things much more clear."* Another participant felt that the experiment made him/her aware of a his/her **lack of basic knowledge**. *"I realized that I have to practice more common logic problems and [understand them well] to not depend too much on AI-based tools."*

Acceptance issues. Besides negative feelings when using the tool, we observed some issues even when not using them. These feelings might be issues with the acceptance of AI-based tools. A participant **preferred to work without help**: *"In the end, I did not use it because I preferred to do it by myself, without help."* Another participant admitted **not liking the tool**.

Issues of the tool. Participants also reported some issues of the tool. One participant reported **not reliable proposals**. *"[The tool] gave me a complete function that did the multiplication. It took only a few seconds. However, the results returned by this function made the vector values converge to around 0.241."* Other participants reported that the tool was **proposing more complex solutions** than those that they would themselves develop. *"In certain moments, it provided good and succinct answers, but, in others, it 'complicated' the problem's solution, using more complicated and unnecessary paths."* Similarly, a participant who had to stop using ChatGPT complained regarding the **difficulty to understand the code** generated by the tool. *"When I stopped using [the tool], I felt a bit lost when the code generated by ChatGPT showed an error, leading me to take more time to understand the code and adjust it."* Another participant who had to start using the tool complained about **changing the train of thought**. *"[The result of the tool] biased my thinking and confused me in the process."*

Limitations of the tool. Some participants also reported some limitations of the tool. For two of them, the tools are **not a substitute for knowledge**. *"A developer is complete on his/her own, with his/her theoretical background (documentation and books), [in contrast] an AI trained with practically random models will never have the same quality of documentation/books/articles and open source codes curated by other developers."* Another issue regards the **need for specific training**. *"I think they help a lot, but one has to know how to use the tool both regarding the input and knowing up to what point it should be used."*. Finally, some participants reported that the tool was not helpful for the exercise.

Positive feelings when not using the tool. Some participants expressed positive feelings when not using the tool. A participant mentioned **feeling good by not depending on AI**:*"I felt good by doing [the task] without the help of AI tools."*

Another participant, who was part of G2, took the task as **challenging not to use**: *"[In the beginning, I was] not very hopeful [about not being able to use AI based tools]. [In the end, I thought:] OK, it will be nice to rely on them."*

5.1.2 *Indifference.*

Some participants expressed **indifference of using or not** the tool since they would have *"done the same manually"*. When asked if they felt dependent on the tool, many participants declared **not feeling dependent** of the tool. As one of them stated: *"the tool generated the solutions I would have done without it, but with the benefit of [a shorter] time."*

5.1.3 *Positive perceptions.*

[3]https://zenodo.org/doi/10.5281/zenodo.10108270

[4]To avoid hurting the blind-review, we do not provide the original answers. If the paper gets accepted, we will update the replication package to contain also the original answers

Value of the tool. In many answers, the participants acknowledge the positive aspects of the tool. For many of them, it was considered **useful**. *"I'm not used to using AI, but there was no problem. I just asked functions to speed up the process, and the answer was always precise."* Some participants were more specific in their answers, explaining how the tool was useful. Two respondents mentioned that the tool gave them a **starting point** for them to perform the task. *"They help a lot and are a good initial step, to help you to have a basis to evolve [the solution]."* Others feel that the tool is **good for repetitive/simple tasks**. *"I think they are good to make simple stuff, but, for more complex things, they get lost a bit while doing the work and we have to fix some stuff. But it saves an initial [repetitive] work, for example, initializing variables."* Another participant mentioned **speeding up the process**. *"[The tools] help a lot in speeding up the process and giving simple solutions to the problems."* Other participants reported that the tool was useful by **augmenting the memory**. *"It helped me understand and remember some concepts and build the code."*

Positive feelings. Several participants expressed positive feelings provoked by the use of the tool. For a participant, the tool increased the **feeling of safety** when facing the problem. *"I felt safe. It optimized my thought process."* Related to that aspect, many respondents mentioned an increasing **feeling of confidence**. *"I felt more confident because, with some right inputs, I can get to the right answer without a huge effort, different from the traditional Google search."* Still, a participant expressed to be **feeling more comfortable** when using the tool: *"When I started using the tool I felt more comfortable, [because I had] the possibility of better expressing what I needed and being able to follow a train of thought based on the answers [the tool] delivers."* A participant mentioned that the experiment made him **feeling not replaceable**. *"It just confirmed that AI tools represent ways to improve developers' productivity but they are not capable of replacing them."* Finally, a participant, still in the pre-test questionnaire, expressed **excitement about the tools' potential**: *"I think technologies like these very interesting, to think about how they work and how they are implemented, their complexity leaves me excited on imagining how I would do something like this."*

5.1.4 Perception of the developer role. Finally, it is interesting to report how the participants perceive the role of a developer. When asked about the perception of what it means to be a developer, four participants stressed the idea of the **developer as a user of tools**. *"A developer has to know how to use the available tools but understand the why and how to use them in the right way."* Another participant stressed the idea of **developers as builders**: *"I still believe that [developers] are builders, from the bureaucratic part until the operational part of the system."* Another one focused on the idea of **developers as information seekers**. *"[The experiment] highlighted something that I implicitly assumed that one of the essential abilities of a developer is to know how to research and collect information."* Finally, a participant focused on **developers as team players**. *"[To be a developer] means to know how to work in a team [to fulfill] a great computing project."*

Table 1: Perception of own skills by group

Group	Number of answers	Perception of own skills		
		Low	Medium	Good
1	10	2	5	3
2	10	1	7	2
3	9	0	6	3
4	8	1	4	3

Table 2: Perception of own experience with AI-based tools by group

Group	Number of answers	Perception of experience		
		No	Low	Good
1	10	2	5	3
2	10	1	8	1
3	9	2	5	2
4	8	2	3	3

Table 3: Answer to the task difficulty by group

Group	Number of answers	Difficulty of the task		
		Easy	Medium	Hard
1	10	1	5	4
2	10	0	9	1
3	9	4	2	3
4	8	1	4	3

5.2 Comparison by groups

To compare the groups, first, we had to compare if their participants were similar regarding skills and experience with AI. To this aim, we compared the answers to the pre-test questionnaire regarding the perceptions of their own skills and experience of AI-based tools by group. This analysis is presented in Tables 1 and 2.

Based on this analysis, we could observe that the groups have similar configurations regarding the perception of their own skills and experience with AI-based tools and the most common is to have a medium perception of the skills and a low experience with AI-based tools.

Then, we analyzed the different groups regarding their reactions to the task. This analysis was based on the answers to the post-questionnaire. One participant from Group 4 did not submit an answer to the post-questionnaire so this analysis was performed based on 37 interviews. First, Table 3 describes how different groups perceived the difficulty of the task.

Table 4 presents the distribution of the answers of the participants who had contact with the tools regarding their perception of AI-based tools.

In these results, we could observe that:

- regarding the impression of AI tools, the comments were balanced on the groups that used AI-based tools in part of the time, but for those who only used the tools, the negative ones were more prevalent;

Table 4: Answers distributions by group and some questions (only groups 2, 3, and 4)

Group	Number	Impression of AI tools			Feeling when starting or stopping the use				Dependence feeling		
		Pos.	Neg.	Not related	Pos.	Neg.	Indiff.	Not related	Pos.	Neg.	Indiff.
2	10	4	5	1	3	4	1	2	0	2	8
3	9	4	5	0	3	3	1	2	0	4	5
4	8	2	6	0	5	3	0	0	0	7	1

- regarding starting or stopping the use, the comments were balanced;
- finally, regarding the feeling of dependence, the negative comments were more prevalent among the participants who used AI the whole time and then among those who had to stop using the tool.

6 DISCUSSION

The qualitative experiment we conducted brought some pieces of evidence that could be related to the first two propositions raised in the social science fiction exercise. First, negative comments related to the feeling of dependence were more common in the groups that worked more with the tools or had to stop using the tool (a similar scenario to that envisioned in the first story), providing evidence related to Proposition 1: **"Developers might feel replaced by AI-based tools."**. Second, negative aspects were more present in the answers from the group that worked the whole time with the tool. This result suggests that Proposition 2: **"AI-based tools might influence the self- perception of developers"** should be further investigated.

In this regard, the codes obtained in the analysis gave interesting insights regarding the reasons for these results. First, developers had the feeling of losing control of the final product, delegating it to the tool. Research has shown that control is an important aspect for developers. The lack of it has been mentioned as a reason for a bad day for a developer [25], and it also influences the adoption of techniques by developers [12, 13].

In this regard, many of the aspects identified in our study indicate potential issues for adopting AI-based tools. Technology acceptance is a topic that has been largely studied [5, 42, 43]. A successful model for this concern has been the Technology Acceptance Model (TAM) [5] and its extension TAM2 [43]. According to the models, the usage of a technology is determined by an intention to use that, in turn, is influenced by the perceived usefulness and perceived ease of use. The perceived usefulness is then influenced by subjective norm, image, job relevance, output quality, and result demonstrability. Many of the codes grouped in the negative aspects, such as not reliable proposals, proposing more complex solutions, and lack of control and trust could be linked to the output quality perceived by developers. TAM and other adoption models have been evaluated to explain the adoption of software methodologies by developers [14, 31]. The results partially supported the theories, describing that the adoption of software methodologies is driven by an organizational mandate to use it, the compatibility of the methodology with the way developers work, and the opinions of developers' coworkers and managers on the technology [31]. However, regarding the acceptance of AI, it has been found that individuals are "mostly driven by subjective norms and emotions" [6].

Although less prevalent, we could also observe, as the participant who felt scared by the tool capacity shows, a certain level of concern about the capabilities of AI to support SE. This aspect is related to what has been called AI anxiety [19], i.e., the anxiety expressed by individuals concerning AI. Li et al. [19] investigated AI anxiety through the lenses of the integrated fear acquisition theory. They identified the following sources of anxiety: privacy violation, biased behavior, job replacement, learning, existential risk, ethics, artificial consciousness, and lack of transparency. This fear humans have towards technology is so present that it is projected in the popular media by depicting robots as monstrous [40].

Finally, the research on AI ethics have focused on the relationship of the software products towards the users [18], and software developers are the users of AI-based tools for SE. Vakkuri et al. [41] proposed a model of the relationship among the key principles in AI ethics. According to the authors, transparency supports the predictability, and both together with fairness generates the trustworthiness on AI. Our results are in line with the model since the many of the negative perceptions we identified, such as lack of trust and lack of control are related to the idea transparency and predictability of the tool.

6.1 Implications for research and practice

Our results represent some implications to research and practice. First, in the development of AI-based tools, that is becoming increasingly relevant both in theory and practice, it is essential to consider the human aspects of the developers, otherwise risking creating distress and hurting the adoption of these tools. These human aspects should be used as input for software companies developing AI based tools which aim to be used by developers.

Second, developers and the companies in which they work should consider potential psychological issues when implementing tools disrupting the way developers work. Our experiment provided indicatives that these psychological issues can vary dramatically from a developer to another.

Finally, SE educators and trainers should work on how to prepare future and current developers to handle new generation of tools. Furthermore, as indicatives in our experimental study have presented, it is important that SE educators assess ways in which to teach new software engineers how to adopt or get used to these tools but are also aware of their limitations.

6.2 Threats to validity

To discuss the potential threats to the validity of our study, we followed the guidelines suggested by Wohlin et al. [45]. The authors present four steps to the validity of experimental results: construct validity, internal validity, external validity, and conclusion validity. They also mention that the counterpart of conclusion validity is reliability for qualitative studies. Since we performed a qualitative experiment, we deemed it necessary to discuss this point as well.

Construct validity. Constructs are concepts that are, in principle, quantifiable but not directly measurable. Construct validity regards whether the constructs in the study are real and if the indicators to infer them really reflect the constructs [29]. Based on the pre-test phase of our study, we analyzed the perceptions of the participants regarding their skills and their experience with the tools. To do so, we relied on their own assessment based on the researchers' analysis. Although this procedure might represent a threat, the potential impact of this issue is limited given that this indicator was just used to compare the groups and not to propose causal effects. It is worth mentioning that during the pilot study, participants were invited to provide feedback regarding any concerns about the pre-and post questionnaire questions.

Internal Validity. We manually analyzed the data to extract the codes used to identify the negative and positive aspects regarding the student's perceptions of the AI-based tools used in our study, which might introduce researcher bias to our results. To mitigate this internal threat in our study, we followed a well-defined analysis procedure in an iterative process with periodic group meetings. The definitions were discussed and refined by the authors. Also, after the iterative coding process, the resulting codes and analysis were discussed among the authors.

External validity. This aspect concerns to what extent our results are generalizable. The generalization of our results may be limited once our population is limited to students and all of them are part of the same R&D environment. Another potential issue may be the proposed treatment that could not represent real situations. To mitigate this threat, we provided supplemental material and encouraged researchers to replicate our study and compare their findings with ours.

Conclusion validity. This aspect regards the relationship between treatment and outcome, i.e., if the observed behavior in the outcome was determined by the treatment and not by other factors not considered. To reduce potential threats in this regard, we compared participants regarding their perceptions of their own skills and experience with AI-based tools. We also compared the results with a control group.

Reliability. This aspect concerns to what extent the results are dependent on the researchers involved in the study. Thus, Merriam [24] suggests checking the consistency of the results and inferences. According to Merriam [24], consistency refers to ensuring that the results consistently follow from the data and no inference cannot be supported after the data analysis. To increase consistency, we performed data analysis when, initially, the first author conducted the coding process alone, and afterward, a second author reviewed the codes alone as well. Finally, all the authors discussed potential issues and fixed the codes accordingly. During this process, we had weekly meetings to discuss and adjust codes and categories until we reached an agreement.

7 CONCLUSIONS

The emergence of AI-based tools will disrupt several aspects of our lives. Knowledge-workers, including software developers, will probably have their jobs profoundly affected by these tools. To explore potential issues in this regard, we performed a social science fiction, creating two futuristic stories involving software engineers. Based on them, we described four propositions regarding possible impacts to human or social aspects of SE. To evaluate two propositions, we conducted a qualitative experiment with students in which they had to solve a programming problem, using or not an AI-based tool. Analyzing the answers of the participants to questionnaires applied before and after the experiment, we collected several perceptions, both positive and negative, of the participants towards the tool.

Our study is just an initial exploration of a potentially huge challenge for SE practice in the next years. It has provided some pieces of evidence that support the idea that the perception of developers of themselves might be impacted and they might fear to be replaced. There are several possibilities to continue explore this research problem. Future work could focus on the evaluation of the presented propositions. For example, once these tools are more widely adopted, controlled experiments could be performed to quantitatively assess the propositions and it would be also possible to better study more social implications, which are harder to emulate in a controlled environment. Our study also demonstrated a possible use for the social science fiction in SE research that has increasingly relied on methods from social sciences.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES). Igor Wiese thanks CNPq #408812/2021-4, MCTIC/CGI/FAPESP #2021/06662-1, Fundação Araucaria and UTFPR. This study was also partially supported by the Ministry of Science, Technology, and Innovations from Brazil, with resources from Law No. 8.248, dated October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex.

REFERENCES

[1] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. 2017. Agile Software Development Methods: Review and Analysis. arXiv:1709.08439 [cs.SE]
[2] Joanna Bryson and Alan Winfield. 2017. Standardizing Ethical Design for Artificial Intelligence and Autonomous Systems. *Computer* 50, 5 (may 2017), 116–119. https://doi.org/10.1109/MC.2017.154
[3] C. Casey. 1995. *Work, Self, and Society: After Industrialism.* Routledge.
[4] Randy Connolly. 2020. Why computing belongs within the social sciences. *Commun. ACM* 63, 8 (jul 2020), 54–59. https://doi.org/10.1145/3383444
[5] Fred D Davis. 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly* 13, 3 (sep 1989), 319. https://doi.org/10.2307/249008
[6] Manlio Del Giudice, Veronica Scuotto, Beatrice Orlando, and Mario Mustilli. 2023. Toward the human – Centered approach. A revised model of individual acceptance of AI. *Human Resource Management Review* 33, 1 (2023), 100856. https://doi.org/10.1016/j.hrmr.2021.100856
[7] Dario Di Nucci, Fabio Palomba, Giuseppe De Rosa, Gabriele Bavota, Rocco Oliveto, and Andrea De Lucia. 2018. A Developer Centered Bug Prediction Model. *IEEE Transactions on Software Engineering* 44, 1 (jan 2018), 5–24. https://doi.org/10.1109/TSE.2017.2659747
[8] Vinicius H. S. Durelli, Rafael S. Durelli, Simone S. Borges, Andre T. Endo, Marcelo M. Eler, Diego R. C. Dias, and Marcelo P. Guimarães. 2019. Machine Learning Applied to Software Testing: A Systematic Mapping Study. *IEEE Transactions*

on Reliability 68, 3 (2019), 1189–1212. https://doi.org/10.1109/TR.2019.2892517
[9] Fabio Ferreira, Luciana Lourdes Silva, and Marco Tulio Valente. 2021. Software engineering meets deep learning: a mapping study. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. Association for Computing Machinery, New York, NY, USA, 1542–1549. https://doi.org/10.1145/3412841.3442029
[10] Neil Gerlach and Sheryl N. Hamilton. 2003. Introduction: A History of Social Science Fiction. *Science Fiction Studies* 30, 2 (2003), 161–173. http://www.jstor.org/stable/4241163
[11] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. 2018. What happens when software developers are (un)happy. *Journal of Systems and Software* 140 (jun 2018), 32–47. https://doi.org/10.1016/j.jss.2018.02.041
[12] Gina Green and Alan R. Hevner. 1999. *Perceived Control of Software Developers and Its Impact on the Successful Diffusion of Information Technology.* Technical Report April. Carnegie Mellon University.
[13] Gina C. Green and Alan R. Hevner. 2000. Successful diffusion of innovations: guidance for software development organizations. *IEEE Software* 17, 6 (2000), 96–103. https://doi.org/10.1109/52.895175
[14] Billc Hardgrave, Fred D Davis, and Cynthia K Riemenschneider. 2003. Investigating Determinants of Software Developers' Intentions to Follow Methodologies. *Journal of Management Information Systems* 20, 1 (2003), 123–151. https://doi.org/10.1080/07421222.2003.11045751
[15] James D. Herbsleb and Deependra Moitra. 2001. Global software development. *IEEE Software* 18, 2 (2001), 16–20. https://doi.org/10.1109/52.914732
[16] Charles Hill, Rachel Bellamy, Thomas Erickson, and Margaret Burnett. 2016. Trials and tribulations of developers of intelligent systems: A field study. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Vol. 2016-Novem. IEEE, 162–170. https://doi.org/10.1109/VLHCC.2016.7739680
[17] Brittany Johnson, Thomas Zimmermann, and Christian Bird. 2021. The Effect of Work Environments on Productivity and Satisfaction of Software Engineers. *IEEE Transactions on Software Engineering* 47, 4 (apr 2021), 736–757. https://doi.org/10.1109/TSE.2019.2903053
[18] Arif Ali Khan, Sher Badshah, Peng Liang, Muhammad Waseem, Bilal Khan, Aakash Ahmad, Mahdi Fahmideh, Mahmood Niazi, and Muhammad Azeem Akbar. 2022. Ethics of AI: A Systematic Literature Review of Principles and Challenges. *The International Conference on Evaluation and Assessment in Software Engineering 2022*, 383–392. https://doi.org/10.1145/3530019.3531329
[19] Jian Li and Jin Song Huang. 2020. Dimensions of artificial intelligence anxiety based on the integrated fear acquisition theory. *Technology in Society* 63 (2020), 101410. https://doi.org/10.1016/j.techsoc.2020.101410
[20] Lucy Ellen Lwakatare, Aiswarya Raj, Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. 2019. A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation. In *Agile Processes in Software Engineering and Extreme Programming*, Philippe Kruchten, Steven Fraser, and François Coallier (Eds.). Springer International Publishing, Cham, 227–243.
[21] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 72 (nov 2019), 23 pages. https://doi.org/10.1145/3359174
[22] Jorge Melegati and Xiaofeng Wang. 2021. Surfacing Paradigms underneath Research on Human and Social Aspects of Software Engineering. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 41–50. https://doi.org/10.1109/CHASE52884.2021.00013
[23] Daniel Méndez Fernández and Jan-Hendrik Passoth. 2019. Empirical software engineering: From discipline to interdiscipline. *Journal of Systems and Software* 148 (feb 2019), 170–179. https://doi.org/10.1016/j.jss.2018.11.019 arXiv:1805.08302
[24] Sharan B Merriam and Elizabeth J Tisdell. 2015. *Qualitative research: A guide to design and implementation.* John Wiley & Sons.
[25] Andre N. Meyer, Earl T. Barr, Christian Bird, and Thomas Zimmermann. 2021. Today Was a Good Day: The Daily Life of Software Developers. *IEEE Transactions on Software Engineering* 47, 5 (2021), 863–880. https://doi.org/10.1109/TSE.2019.2904957
[26] N. Montfort. 2017. *The Future.* MIT Press.
[27] Nicole Novielli and Alexander Serebrenik. 2019. Sentiment and Emotion in Software Engineering. *IEEE Software* 36, 5 (sep 2019), 6–23. https://doi.org/10.1109/MS.2019.2924013
[28] Jonhnanthan Oliveira, Rohit Gheyi, Melina Mongiovi, Gustavo Soares, Márcio Ribeiro, and Alessandro Garcia. 2019. Revisiting the refactoring mechanics. *Information and Software Technology* 110, November 2018 (2019), 136–138. https://doi.org/10.1016/j.infsof.2019.03.002
[29] Paul Ralph and Ewan Tempero. 2018. Construct Validity in Software Engineering Research and Software Metrics. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, Vol. Part F1377. ACM, New York, NY, USA, 13–23. https://doi.org/10.1145/3210459.3210461
[30] Tom Redshaw. 2020. What Is Digital Society? Reflections on the Aims and Purpose of Digital Sociology. *Sociology* 54, 2 (apr 2020), 425–431. https://doi.org/10.1177/0038038519880114
[31] Cynthia K. Riemenschneider, Bill C. Hardgrave, and Fred D. Davis. 2002. Explaining software developer acceptance of methodologies: A comparison of five theoretical models. *IEEE Transactions on Software Engineering* 28, 12 (2002), 1135–1145. https://doi.org/10.1109/TSE.2002.1158287
[32] Sue Robinson and Andrew L. Mendelson. 2012. A Qualitative Experiment: Research on Mediated Meaning Construction Using a Hybrid Approach. *Journal of Mixed Methods Research* 6, 4 (oct 2012), 332–347. https://doi.org/10.1177/1558689812444789
[33] Sharon Ryan and Rory V. O'Connor. 2013. Acquiring and sharing tacit knowledge in software development teams: An empirical study. *Information and Software Technology* 55, 9 (sep 2013), 1614–1624. https://doi.org/10.1016/j.infsof.2013.02.013
[34] Ken Schwaber and Jeff Sutherland. 2012. The Scrum Guide. https://api.semanticscholar.org/CorpusID:114128971
[35] Neil Selwyn, Luci Pangrazio, Selena Nemorin, and Carlo Perrotta. 2020. What might the school of 2030 be like? An exercise in social science fiction. *Learning, Media and Technology* 45, 1 (2020), 90–106. https://doi.org/10.1080/17439884.2020.1694944
[36] Gaurav G. Sharma and Klaas-Jan Stol. 2020. Exploring onboarding success, organizational fit, and turnover intention of software professionals. *Journal of Systems and Software* 159 (jan 2020), 110442. https://doi.org/10.1016/j.jss.2019.110442
[37] Mary Shaw. 1990. Prospects for an engineering discipline of software. *IEEE Software* 7, 6 (nov 1990), 15–24. https://doi.org/10.1109/52.60586
[38] Mary Shaw. 2009. Continuing prospects for an engineering discipline of software. *IEEE Software* 26, 6 (2009), 64–67. https://doi.org/10.1109/MS.2009.172
[39] Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. 2020. IntelliCode compose: code generation using transformer. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, New York, NY, USA, 1433–1443. https://doi.org/10.1145/3368089.3417058 arXiv:2005.08025
[40] Michael Szollosy. 2017. Freud, Frankenstein and our fear of robots: projection in our cultural perception of technology. *AI and Society* 32, 3 (2017), 433–439. https://doi.org/10.1007/s00146-016-0654-7
[41] Ville Vakkuri, Kai-Kristian Kemell, Joni Kultanen, and Pekka Abrahamsson. 2020. The Current State of Industrial Practice in Artificial Intelligence Ethics. *IEEE Software* 37, 4 (jul 2020), 50–57. https://doi.org/10.1109/MS.2020.2985621
[42] Venkatesh, Morris, Davis, and Davis. 2003. User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly* 27, 3 (2003), 425. https://doi.org/10.2307/30036540
[43] Viswanath Venkatesh and Fred D. Davis. 2000. A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management Science* 46, 2 (feb 2000), 186–204. https://doi.org/10.1287/mnsc.46.2.186.11926
[44] Jess Whittlestone, Rune Nyrup, Anna Alexandrova, and Stephen Cave. 2019. The Role and Limits of Principles in AI Ethics: Towards a Focus on Tensions. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (Honolulu, HI, USA) *(AIES '19)*. Association for Computing Machinery, New York, NY, USA, 195–200. https://doi.org/10.1145/3306618.3314289
[45] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. Planning. In *Experimentation in Software Engineering*. Vol. 9783642290. Springer Berlin Heidelberg, Berlin, Heidelberg, 89–116. https://doi.org/10.1007/978-3-642-29044-2_8
[46] Kareshna Zamani, Didar Zowghi, and Chetan Arora. 2021. Machine Learning in Requirements Engineering: A Mapping Study. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. 116–125. https://doi.org/10.1109/REW53955.2021.00023

A QUESTIONNAIRES

Below, we present the questionnaires applied during the qualitative experiment. First, the pre-experiment questionnaire, answered by all participants, is as follows:

(1) What is your participant ID?
(2) In your opinion, what does it mean to be a software developer?
(3) What is your perception regarding your own programming skills? For example, do you consider yourself to be a good programmer? etc.
(4) What is your experience with generative AI based software engineering tools? For example, Github Copilot. Do you use them in your daily routine? If so, which ones?

(5) What is your opinion of generative AI based software engineering tools? For example, Github Copilot.

Below, we presented the two forms applied after the experiment. First, the questionnaire answered by the participants who were assigned to the control.

(1) What is your participant ID?
(2) How difficult was the problem and why?
(3) Has the experiment altered your perception of what it means to be a software developer?

Finally, the questionnaire below was answered by all the participants who had used the tool.

(1) What is your participant ID?
(2) How difficult was the problem and why?
(3) What was your impression of the generative AI-based tools you have used?
(4) How did you feel once you had to stop (or begin) using the generative AI-based tool?
(5) Did you feel dependent on the tool?
(6) Have your opinions on generative AI-based tools changed after the experiment? If so, how?
(7) Has the experiment altered your perception of what it means to be a software developer?