Human and Machine: How Software Engineers Perceive and Engage with AI-Assisted Code Reviews Compared to Their Peers

1st Adam Alami
Mærsk Mc-Kinney Møller Institute
University of Southern Denmark

2nd Neil Ernst
Department of Computer Science
University of Victoria

***Abstract*—The integration of artificial intelligence (AI) continues to increase and evolve, including in software engineering (SE). This integration involves processes traditionally entrusted to humans, such as coding. However, the impact on socio-technical processes like code review remains underexplored. In this interview-based study (20 interviewees), we investigate how software engineers perceive and engage with Large Language Model (LLM)-assisted code reviews compared to human peer-led reviews. In this inherently human-centric process, we aim to understand how software engineers navigate the introduction of AI into collaborative workflows. We found that engagement in code review is multi-dimensional, spanning cognitive, emotional, and behavioral dimensions. The introduction of LLM-assisted review impacts some of these attributes. For example, there is less need for emotional regulation and coping mechanisms when dealing with an LLM compared to peers. However, the cognitive load sometimes is higher in dealing with LLM-generated feedback due to its excessive details. Software engineers use a similar sense-making process to evaluate and adopt feedback suggestions from their peers and the LLM. However, the LLM feedback adoption is constrained by trust and lack of context in the review. Our findings contribute to a deeper understanding of how AI tools are impacting SE socio-technical processes and provide insights into the future of AI-human collaboration in SE practices.**

***Index Terms*—Code Review, Large Language Models, LLM-supported software engineering, Human-AI collaboration**

I. Introduction

The integration of artificial intelligence (AI) is growing in many processes, including software engineering (SE) [24]. However, for decades and despite various practices being automated, SE remains a human activity. The integration of AI into a socio-technical process such as SE, historically dominated by human control, interactions, and decision-making, may imply a paradigm shift in the way software is engineered.

Research and products aiming to leverage AI for software engineering are targeting different core SE activities. Research efforts have shown interest in requirements engineering, e.g., [22], [54], [62], software design, e.g., [34], [66], construction, e.g., [48], [80] and testing, e.g., [68], [81].

Machine learning (ML) technologies have opened up the possibility to automate difficult SE tasks, traditionally entrusted to humans. Advances in neural network architectures, such as recurrent neural networks and transformers [78], have been used to advance the state-of-the-art for many difficult automated software engineering tasks [21], [24]. These technologies, among others, have contributed to an array of commercial products such as Tabnine [3], CodeX [2], Github's Copilot [1] and ChatGPT.

While these products are task specific, i.e., AI pair programming, Large Language Models (LLMs) like GPT-4 have emerged as powerful multi-task products, capable of not only generating code but also assisting in other software development activities, including documentation, debugging, design suggestions, and code review [24]. The versatility and broad applicability of LLMs have the potential to fundamentally alter the landscape of software engineering by either reducing the reliance on human intervention or support in tasks that were previously considered too complex for automation [76], [82].

Although the products landscape designated to LLM-supported SE and the enthusiasm for their adoption are increasing [4], [24], our understanding of the impact of this shift on the social and human dynamics within SE environments remains limited [10]. SE is inherently a socio-technical practice [12]. The engineering process is entwined with collaboration, human judgment, emotional interactions, and decision-making [38]. As we witness a potential paradigm shift—moving from human-dominated processes to AI-supported workflows—it is important to investigate how these changes affect the human and social elements of SE.

Most of human SE activities are collaborative. Thus, the integration of AI into the already complex human and social workflows necessitates a thorough understanding of the implications. Specifically, the introduction of AI in practices traditionally governed by human expertise raises questions about the human-AI engagement and behavioral responses of practitioners like software engineers. For example, Malone et al. suggest that the most challenging changes in AI integration are not computers replacing humans but rather people and computers working together as integrated "superminds" [52]. To effectively harness the potential of these "superminds" [52], it is crucial to understand how software engineers engage with and respond to AI tools in their workflows. Therefore, we propose:

RQ: How do software engineers perceive and engage with

2574-1837/25/$31.00 ©2025 IEEE
DOI 10.1109/CHASE66643.2025.00016

LLM-assisted code reviews compared to their peers?

Engagement, in the context of our study, is the ways and the extent to which software engineers actively interact with, respond to, and incorporate feedback from the review process. We opted for code review as a SE process, because it is inherently a human-centric process that relies on collaboration, judgment, and communication. This makes it ideal to evaluate how engineers experience the engagement in an LLM-supported alternative to a process traditionally characterized by human-intensive interactions.

To investigate our **RQ**, we set an interview study (20 interviewees) anchored in tangible review experiences. Prior to the interviews, we asked our participants to submit a sample of their own authored code. Then, we assigned two to three human reviewers to every author and asked them to submit written reviews. We used the reviews during the interviews to prompt our interviewees to reflect on their experiences, perceptions, and emotional reactions using recent and concrete experiences. In the second part of the interview, we shared an LLM-generated review of their authored code to explore how their reactions and engagement differ when interacting with AI-generated feedback as opposed to human-generated feedback.

By juxtaposing these insights, we aimed to understand and capture any deviations in engagement in the case of LLM-assisted reviews compared to peers. This allowed us to identify challenges and constraints in the integration of AI tools into collaborative SE processes. This understanding may inform the development of more effective and harmonious AI-human collaborative workflows in the future.

We contribute to AI integration in SE literature by identifying the dimensions of engagement in code review, which include cognitive, emotional, and behavioral responses to feedback, and how the introduction of LLMs could influence hese engagement attributes.

II. Related Work

While code review is part of our research context, it is not the primary focus of our investigation. We used code review as a lens through which we evaluated a broader and critical issue of how software engineers perceive and engage with AI tools in a socio-technical process. In particular, we examine how engineers respond to AI-generated feedback compared to feedback from peers. Therefore, the scope of our related work is to draw on studies focusing on human-AI collaboration and how software engineers engage with AI in SE processes. We set our results in the context of existing literature, including code reviews, in a discussion section (§V).

Human-tool integration before the advent of AI (i.e., LLM-based assistants) often took the form of reporting automated build and test results or other static analysis checks (e.g., linting or security scans). A simple example is the output of a compilation run and the resulting error messages. Even this relatively simple set of tool results can be difficult for humans to understand and work with [23]. Static analysis results might take the form of reporting that a particular code file contains known security issues (e.g., possible SQL injections). This is a considerably more actionable and explicable report than the detailed AI-generated code reviews we consider in this paper. But even these simpler reports are challenging to integrate into human decision-making. For example, Johnson et al. showed that reporting the factual outputs of the analysis is not sufficient for developers to adopt a static analysis tool [35].

A common medium for human-tool integration is a bot, typically serving as a textual/chat interface to the underlying static analyzer. For example, Repairnator was a bot for GitHub that was an interface to a sophisticated program repair tool [58]. The bot's designers found that without *explanation* it was hard or impossible to get humans to accept the changes [57].

In a wider study, researchers examined bot trust and autonomy and found that individuals have varying levels of trust in the bot's recommendations [26]. That paper recommended that bot interaction characteristics should be customizable. In addition, *how the bot appears* (i.e., avatar, language, name) can have a big impact on human acceptance: merely revealing that something is a bot can dramatically change human perceptions [60].

These perceptions may be changing as LLM capabilities (and surrounding hype) make AI assistance more commonplace. Nonetheless, Al Haque et al. have identified several challenges, including that the AI-generated content was found to be overly polite, overly detailed, and lacked trustworthiness [5]. To date, most of the work on LLMs and coding, including code review, has focused on the technical aspects of LLMs rather than human interaction issues. Technical work looks at creating datasets of pull requests and code problems [46], examining how to improve fine-tuning, e.g., LLaMA-Reviewer [50], and exploring different architectures for the underlying neural networks [77].

Lack of specific context is a challenge for AI tools. Developers get annoyed when the AI does not understand the locality of their codebase, although for boilerplate or generic code, developers appreciate saving keystrokes [47]. As Panichella et al. point out [64], since code review tasks themselves are frequently changing, e.g., as the organization develops a new understanding of concurrency, statically trained LLMs may miss evolving changes.

In sum, these studies show that achieving effective human-tool integration is about much more than functionality [5], [46], [50]. Tool design must also address human aspects such as trust, interpretability, and engineers' preferences for communication style and content [26], [60]. Our study contributes to this body of work by identifying potential shifts in software engineers engagement in response to AI-driven code review.

III. Methods

We opted for an interview study to investigate our **RQ** objectives. This choice is better suited when the phenomenon under study requires uncovering complex social and behavioral nuances that influence human judgment and interactions in socio-technical processes [12], [43]. Interviews are also a good

tool to access perspectives and insights ingrained deeply in personal beliefs, attitudes, and behaviors [44], [70]. By tapping into these core personal and deep-seated cognitive constructs [44], [70], we aim to collect data in line with our study objectives.

We designed a research process to capture real-life practices, yet within the constraints inherent to a research environment. For example, our participants remained anonymous to each other, opposed to a professional context where authors and reviewers are known to each other. Upon the completion of our recruitment and selection process, we asked our participants to submit a code they had authored. Then, we assigned to each author two to three reviewers (drawn from other participants in the study) and asked them to submit a written review as if they were to conduct a peer review in their professional contexts. Prior to the interviews, we shared the peer reviews with the authors. We asked the participants permission to prompt ChatGPT to review their code and inform them that we will share the LLM-generated reviews in the interview for discussion.

In the first part of the interview, we used the reviews as catalysts to prompt the engineers to explain and elaborate further on their responses to our questions. In the second part, we shared an LLM-generated review of the same code authored by the interviewees to prompt them to share their attitudes and understand how they would engage with it compared to human-authored reviews. This approach ensures that we capture the nuanced differences in how software engineers engage with and respond to both human and LLM-generated feedback. First, we grounded the discussion in their real-world practices and the study's review experience. Then, we prompted them to contrast their earlier claims when reviews are AI-generated. This dual-method strategy, combining data in both engagement with peer-conducted and AI-generated reviews, aligns with the study objectives. The process was explained to participants during the recruitment.

A. Interviewee recruitment & selection

To recruit our interviewees, we used Prolific[1], a research market platform. Given that the platform does not verify nor evaluate self-reported skills [8], we carried out a pre-screening process to qualify our potential participant, following the guidelines suggested by Alami and colleagues [8].

Alami et al. recommend using an iterative and controlled prescreening process, using task-oriented questions that go beyond theoretical understanding to help filter genuine engineers from those who may rely on external resources, such as Googling or prompting an LLM for answers [8].

In the pre-screening survey, we used a programming task, a critical-thinking question, and a prompt for participants to share a problem-solving scenario from their own experience. We iteratively and manually assessed the survey answers [8]. First, we evaluated the answers for AI-generated content using ChatGPT 4.0. Subsequently, we manually evaluate the content's quality and coherence [8]. We capped the number of participants to 500, and after the prescreening process, we ended up with 353 qualified engineers. Then, we ran a final pre-selection survey where we asked for further demographic data, programming language skills, and whether the participants were willing to participate in an interview study. We limited this pre-selection to 100, and 76 agreed to participate in the interviews. This 2-phased approach had different objectives. While in the initial prescreening, we focused on evaluating potential participants' skills, i.e., are they genuinely software engineers [8], in the second pre-selection, we wanted to know whether those qualified software engineers are willing to take part in an interview study and the programming languages they may submit their code and review other participants' codes. We paid £0,50 for the initial prescreening and pre-selection surveys and £60,00 for the participation in the interview. The selection process took place in the first and second weeks of August 2024.

Table I documents our participants' characteristics, providing their current roles, experience levels, gender, industry sectors, and the programming languages they opted to submit their code in. The column "Reviewers" lists the reviewers assigned to each participant. We managed to assign to each participant a minimum of two reviews. Even though we aimed for three reviews per participant to align with industry and open-source community practices [6], [7], we did not manage to meet this expectation due to the varying programming skills and language preferences among participants, which limited the pool of suitable reviewers for certain code submissions. In assigning reviewers to authors, we aimed for a diverse range of experiences and backgrounds. For example, P1, a senior software engineer with an experience of 6–10 years, was assigned reviewers with experiences ranging from less than two years to ten years. This diversity in reviewers' experience levels was intentionally designed to enrich our data with varying levels of expertise and perspectives and feedback from both novice and seasoned reviewers. We also aimed to be diverse in gender and country in the reviewers selection. For example, two females and one male from two different countries reviewed P18.

B. Data collection

Our research design sought to ground the data collected during the interview in concrete experiences by our participants. By anchoring the interview discussions in reviews received from other participants, we grounded the data in personal, recent, and close experiences. To this end, prior to the interviews, we shared the reviews with all participants and asked them to read and reflect on them in preparation for the interview.

The interview consisted of two parts. In the first part, we focused on the engagement with human peer's reviews, and in the second part, we shared a ChatGPT 4.0-generated review with the participants and asked them to share their perceptions and reactions to the LLM-generated feedback on their code. We opted for ChatGPT for its wide accessibility and familiarity

[1] https://www.prolific.co/

TABLE I
INTERVIEWEES' CHARACHTERESTICS.

#	Role	Exp.	Gender	Industry sector	Language	Reviewers	Country
P1	Sr. Software Engineer	6-10 years	Male	IT Services	JavaScript	P10, P19, & P20	Germany
P2	Sr. Software Engineer	6-10 years	Male	Consulting	Java	P1, P3, & P19	UK
P3	Sr. Software Engineer	>10 years	Male	Telecommunication	Python	P4, P12, & P18	UK
P4	Software Engineer	3-5 years	Male	Telecommunication	Python	P3, P12, & P20	Germany
P5	Software Engineer	>10 years	Male	IT Services	C#	P11 & P17	USA
P6	Software Engineer	6-10 years	Non-binary/third gender	Finance	Java	P6, P11, & P14	Netherlands
P7	Tech Lead	>10 years	Male/	Finance	Java	P11 & P14	Canada
P8	Software Engineer	3-5 years	Male	Aviation	Python	P9, P17, & P20	Spain
P9	DevOps Engineer	>10 years	Female	Health Care	Python	P8, P13, & P20	UK
P10	Software Engineer	1-2 years	Male	IT Services	Python	P3, P12, & P18	UK
P11	Software Engineer	3-5 years	Female	Finance	Java	P6 & P7	South Africa
P12	Sr. Software Engineer	>10 years	Female	Finance	Python	P4, P12, & P19	UK
P13	Sr. Software Engineer	6-10 years	Male	IT Services	TypeScript	P16 & P17	Ireland
P14	Software Engineer	6-10 years	Male	IT Services	Java	P5, P8, & P13	Portugal
P15	Sr. Software Engineer	3-5 years	Female	Government Services	Python	P16 & P18	Italy
P16	Software Engineer	1-2 years	Male	IT Services	JavaScript	P6, 13, & P21	Portugal
P17	Software Engineer	3-5 years	Male	Media & Entertainment	Python	P5, P8, & P9	Uk
P18	Software Engineer	6-10 years	Male	IT Services	C++	P2, P12, & P15	UK
P19	Sr. Software Engineer	6-10 years	Female	IT Services	Java	P2 & P10	Germany
P20	Software Engineer	3-5 years	Non-binary/third gender	IT Services	JavaScript	P1 & P10	Ireland

TABLE II
KEY PARTS OF THE INTERVIEW GUIDE

Introduction
Can you please introduce yourself? (Include educational background, current role, and years of experience in software development.)
Section I: Approach to Reviewing Code
Are there any coding standards or best practices that you kept in mind during the reviews we assigned to you?
What did you consider most when you wrote your feedback for the assigned code?
Section II: Engagement with peers' feedback on authored code
How do you typically feel when you receive feedback on your code from peers?
In the feedback you received from the other participants, what elements of the feedback do you find most useful or valuable? And why?
Section III: Engagement with LLM-generated review
How did you feel when you first read the LLM-generated review of your code?
Would you respond and react to the LLM-generated feedback the same way as you do to human's feedback?
Conclusion
Based on your experiences in this study, how would you summarize the main differences between peer and LLM-generated code reviews?
Is there anything else you would like to share about your experience in this study or with code reviews in general?

among a diverse audience. To generate ChatGPT reviews, we used this prompt: "You are an expert of [the programming language]. Provide a thorough review of the attached code." This prompt design aimed to generate expert-level feedback, relatable to what a human reviewer might offer [16]. Such feedback is what developers would expect from knowledgeable peers [16]. The intent of this prompt design is also to create a realistic basis for comparing human and AI-generated reviews. For example, a prompt that lacks specificity or context may generate unrealistic and unrelatable feedback [49].

Our interviews were semi-structured. Although we designed an interview guide, its purpose was to guide the conversation while allowing fluidity by prompting the interviewee to elaborate and explain further based on their responses. We also used examples from the reviews they received to engage them in deeper reflection of how they perceived the comments they received either from the LLM or other participants. Table II illustrates examples of the questions we asked. The full guide is available in the shared documents package (Sect. III-E).

All interviews were conducted using Zoom and lasted 40-60 minutes, with a total of 17h12min of audio. The audio generated a total of 271 pages verbatim after transcribing, an average of approximately 14 pages per interview. The first author conducted the interviews in the first and second weeks of September 2024. We used Otter.ai[2] an online transcription tool to transcribe the audio recordings.

C. Data analysis

We adopted a constructivist stance [36], which aligns with our objective to understand software engineers' perceptions and experiences with AI-generated feedback compared to their peers. By adopting this stance, we recognize that knowledge and meaning are actively constructed by individuals through their experiences and interactions [36].

We followed Miles et al. [55] recommendations for the analysis process. First, we conducted a preliminary *First Cycle* analysis. In this early stage of the analysis, we selected "chunks" of data that are pertinent to our RQ and assigned them codes [55]. Using an inductive strategy, this condensing yet analytical exercise allowed us to reduce the data corps into meaningful concepts [56].

In the *Second Cycle* that followed, we synthesized all the *First Cycle* codes into "Pattern codes" [55]. In this integrative activity, we looked for patterns across the previous phase codes by identifying recurring themes that linked different codes together; either have similarities, form a process, or their relations form a cohesive construct by complementing each other [55]. This process allowed us to develop higher-level constructs and refine our understanding of the underlying phenomena.

Table III documents an example of some of our Pattern Codes and their corresponding *First Cycle* codes. Each pattern code was counted once per interview, regardless of the number

[2]https://otter.ai/

TABLE III
EXAMPLE OF PATTERN CODES AND THEIR CORRESPONDING FIRST CYCLE CODES

Pattern codes	First Cycle codes	Examples from the data
Reviewer context	Peers' seniority	*"... if it's a junior developer, I just have more suggestions about things that I spot in the code. Where is if it's a senior developer, sometimes I won't have as many suggestions, and I'll just let them know maybe one or two things. But other than that, I'll just let them know looks good to me, and I'll give it a thumbs up for approval"* (P5).
	Familiarity with peers	*"I know the team very well, the team knows me very well ... when you're in that kind of environment, you kind of, you know, you get straight to the point, because, you know, you don't want to, you don't want to waste their time ... I don't really put the positive feedback, because it was like, you know, they probably, you know that it's good"* (P2).
	Trust in LLMs	*"... with chatgpt, I can't be like I would 100% validate ... I would not go like, just blindly trust it. And with my peers, I would at least have a general idea of, like, how much knowledge they have and how, how well they are delivering feedback"* (P4).
Engagement	Cognitive	*"... [The review received from] number six [i.e., P6] was the most, easiest for me to digest, and I felt like the tone wasn't really too harsh, or I think it was very professional"* (P7).
	Emotional	*"It feels nice [to receive helpful feedback]. It feels like he genuinely cares for the code I produce, and he wants to help me do well"* (P10).
	Behavioral	*"I think it [peer reviews] help to improve myself. So always even be a better version of myself"* (P15).

of times it was mentioned within a particular interview. The final column provides examples from the data.

The first and second coding cycles were conducted by the first author. Then, iteratively, the second author reviewed the codes and Pattern Codes, provided comments, and proposed new and alternative codes, until a consensus was reached. The first author then revised and offered a final set of codes and Pattern Codes. This "reliability check" [20], [55], allowed us to validate our coding judgments and settle our discrepancies, resulting in more trustworthy interpretations.

We monitored data *saturation* [9], [59] throughout our analytical process. We commenced the analysis as soon as the interview transcripts became available. Then, iteratively, as data and the analytical outputs became available, we compared data and emerging themes to assess saturation points [59]. By switching back and forth between data collection and analysis [15], we managed to observe when our Pattern Codes reoccur strongly in the data, hence reaching saturation. We documented this process in our shared documents package.

Upon the completion of the analysis, we organized a *member checking* [14] activity to collect feedback from our interviewees on our findings (see Sect. VI for further details).

D. Ethics and informed consent

All interviewees were asked to read a formal consent request on the first page of the pre-selection survey and asked to agree or disagree to participate. The consent covered anonymized data sharing in public repositories and manuscripts, the voluntary nature of participation, withdrawal from the study, the procedures for ensuring confidentiality, and other ethical considerations as per the authors' university policies. Ethical approvals, as per the authors' university requirements, were obtained prior to the study commencing.

E. Shared documents package

In this package[3], we share code snippets and reviews submitted by all participants, the interview pre-screening questionnaire, ChatGPT-generated reviews, member checking questionnaire and responses, and the interview guide. The package can be found here.[4]

IV. FINDINGS

Recall, our **RQ** seeks to understand engagement in code review and how it may be impacted if we introduce LLM-assisted reviews. Engagement, in the context of our study, is the ways and the extent to which software engineers actively interact with, respond to, and incorporate feedback from the review process. **We found that engagement is multi-dimensional, spanning *cognitive*, *emotional*, and *behavioral* responses**. While experiencing these responses, software engineers undertake a *sense-making process* of the feedback in order to make a decision regarding its adoption. We also found that *human-AI collaboration* in the context of code review is characterized by divergence in preferences regarding how feedback should be delivered.

Engagement is initiated by receiving feedback from peers or the LLM (in the case of our study), which has two key components, *content* and *delivery*. Both influence how engineers become involved emotionally and cognitively with the feedback. For example, "harsh" (e.g., P17) and "toxic" (e.g., P2, P6, and P10) feedback evoke negative emotions, which in response prompt emotional regulation from the engineers as a coping mechanism. However, this emotional regulation becomes less relevant in the case of LLM-assisted review, as these tools use a positive and professional tone.

[3]Interview transcripts will be published on acceptance.
[4]https://doi.org/10.5281/zenodo.14000259.

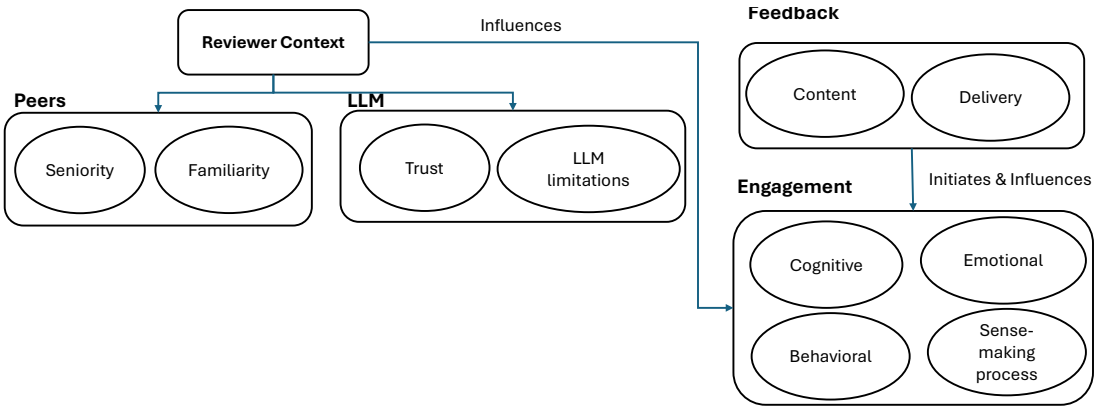

Fig. 1. An abstract presentation of the findings, capturing engagement and how engineers respond to feedback. The line from **Reviewer Context** to **Peer** and **LLM** indicates two type of reviewers, and the ovals inside the rectangles are elements of the reviewer contexts.

Engagement is also influenced by the *reviewer context*. For peers, it is their seniority and familiarity with each other. For LLMs, it is the engineer's trust in LLMs capabilities and other inherent limitations, such as a lack of context relevant to the codebase. Figure 1 captures this dynamic. Below we elaborate on the themes of our findings.

A. Feedback

Feedback is the crux of a code review. It shapes the engagement process. Both the content and the delivery style determine the cognitive effort required by engineers; however, delivery style triggers more emotional and behavioral responses, either negative or positive.

For authors, the cognitive load appears to be low when the feedback is constructive or precise. For example, when P16 was asked to explain the rationale behind his direct and precise comments, he explained: *"... you have to be direct and precise, otherwise people will not understand. But the more direct and logical, I guess it's easier for the programmer to understand what to do"* (P16).

When P8 was asked to explain his tone style in commenting on P9's code, he said: *"... when I get code reviews that are like, super picky ... I think, like, the intention that you get from the code review makes you feel one way or another and makes you perform better or worse ... I won't be in the mood like, let's try to do it ... But if I get, like, a review that ... more constructive ... I will go try to do it my best ... open to more improvement"* (P8). This account shows resignation as an emotional response when the feedback is *"super picky"* and comfort when *"constructive."*

The feeling of resignation or being out of the mood is demotivating for P8, *"I won't be in the mood like, let's try to do it"* (P8). On the other hand, the feeling of comfort seems to have a contrasting effect: *"I will try to do it my best"* (P8).

The introduction of LLM-assisted review showed that the cognitive load required to process LLM reviews seems higher; however, the emotional response appears to be less taxing compared to peers'. When P14 was asked to compare the comments from other participants and the LLM, he commented: *"ChatGPT review will take more time and effort to analyze and review, and others are more straightforward; they identify the problem and suggest a solution for it. This [ChatGPT-generated review] will take more time and mental effort to analyze"* (P14). However, this could be improved with further fine-tuning of the prompt and training of future models based on a foundational model like ChatGPT.

Emotionally, engagement with LLM-generated reviews seems less taxing for our interviewees. When P8 was prompted to explain why he prefers ChatGPT reviews over some of the other participants, he explained: *"... ChatGPT is also polite. So as long as it's polite, I don't really mind ... but a person may not be polite. So, yeah, I wouldn't mind [replacing peers with an LLM]. ChatGPT would never be picky"* (P8). This preference for ChatGPT seems to be rooted in its lack of emotional variability as opposed to humans, who can be "not polite" and "picky." ChatGPT maintains a consistent positive tone, reducing the likelihood of P8 experiencing a taxing emotional experience during the review process. This was echoed in P10 answer to the same question, i.e., why preferring LLM, *"I don't think it [ChatGPT] ever will be harsh ... it just made it easier to kind of accept what it was telling me"* (P10).

Feedback, whether constructive or overly critical, elicits varying emotional and behavioral responses, from resignation to motivation. Our data show that clear and constructive feedback tends to reduce cognitive load and fosters a sense of improvement for the code. The introduction of LLM-assisted reviews seems to offer a less emotionally taxing experience with their consistent positive tone, though potentially at the cost of increased cognitive effort.

B. Engagement

While experiencing their *cognitive*, *emotional*, and *behavioral* responses to the feedback, software engineers engage in a *sense-making process* to make a decision regarding its adoption.

Cognitive: The cognitive part of engagement refers to the mental resources and effort required to process feedback or review code. As reviewers, some engineers in our sample claim that reviewing large code changes is mentally demanding. P2 explains: *"I like code review; when the tickets are small ... if*

they're doing big changes, your brain gets a bit tired ... sloppy, and you won't be able to catch everything" (P2).

Our participants assumed the roles of code authors and reviewers for other participants' code and we designed the interview guide to prompt them about both roles. As authors, our participants claim that the cognitive load is lower when the feedback from peers is constructive or precise. P7 described the review he received from P6 *"easy to digest"*, because he *"felt"* that the tone was not *"too harsh"* and *"professional."* P7's experience demonstrates more efficient cognitive engagement; P6's feedback mitigated high cognitive load. The clarity and professionalism of P6's feedback made it easier for P7 to process, reducing his mental effort required to engage with it. This may suggest that feedback delivered in a constructive and respectful manner fosters both positive emotional responses and facilitates more efficient cognitive engagement.

Compared to LLM-assisted review, the cognitive load required to process the review is higher than that of peers' reviews. P14 states: *"ChatGPT review will take more time and effort to analyze and to review, and others [participant's reviews] are more straightforward"* (P14).

Emotional: Emotional engagement is more prevalent when engineers assume the role of authors. Reviewers are not always aware of the impact of their delivery styles. For instance, when we asked P12 and P18 to clarify their intention to maintain a positive tone in their comments, P12 responded, *"I think I don't know whether it's something that I do consciously,"* and P18, *"I try to be as much as possible neutral,"* respectively.

When processing feedback, engineers may experience a range of positive and negative emotions, contingent on the content and the delivery styles used by reviewers. For example, P2 expressed a feeling of "inspiration" and support in response to P19's constructive and invested feedback, *"I was overwhelmed with P19 I liked the way that they formulated the code review. It was easy to read. I liked the way that they I think for me ... It inspires me sometimes with some of the colleagues, they go above and beyond"* (P2).

However, P6 had a different emotional experience with P11's feedback, as he perceived it as *"brutal"*, causing a perceived feeling of an *"attack."* He explained: *"... there were some feedback where I felt like, just being very brutal ... I do think that there was a tone that was a bit like off and brutal ... I just felt like it was kind of like attacking me"* (P6).

To alleviate some of the emotional taxation of negative feedback, engineers resort to emotional regulation or draw upon their personal values. Some of the emotional regulation strategies reported in our data include consciously avoiding negative reactions (e.g., P4), "to not taking it personally" (e.g., P10, P11), and mindfulness (e.g., P12). Software engineers also turn to their personal values to guide their behavior to navigate challenging feedback situations. Some of these values are "avoiding hostility" (e.g., P10), avoiding "aggression" (e.g., P16), and non-confrontational approach (e.g., P5, P7, P11). While emotional regulations are strategies used to manage individual emotional responses, personal values are guiding principles that shape long-term behavior and decision-making during challenging situations [29], [69].

Compared to LLM-assisted review, the engagement is less taxing emotionally than that of peers, as previously mentioned. P6 explains: *"I feel like a tone goes a long way as well in terms of, like how you take feedback ... ChatGPT in that regards is better. I reacted positively to its tone"* (P6).

Behavioral: While emotional engagement involves the internal feelings and affective responses elicited by feedback, behavioral engagement refers to the observable actions of individuals in response to feedback as a social stimulus [25]. It encompasses how software engineers respond to feedback, including their willingness to seek clarification and proactively implement it. We assessed behavioral engagement in our data through expressed actions like initiating further dialogue with the reviewers or implementing changes to code [37], [73].

In reference to P2 and P6's experiences above, their emotional responses to feedback influenced their subsequent behaviors. P2's behavioral engagement seems generally positive, marked by reflection and increased effort. P6's engagement appeared also positive, yet mixed, wanting more clarification and showing defensiveness. P2's actions show steps taken to revise and reflect on the code as a response to P19 feedback. He explained: *"I went back to the code, and then I looked at the feedback, and then I looked back at the code thinking, okay, I see what they mean now"* (P2). This behavior may be more conducive to constructive improvement of the code, e.g., *"you're inspired to do well"* (P2). The absence of negative emotional impact allowed P2 to focus directly on the content rather than its delivery.

While P6 still engaged positively by seeking clarifications, his engagement may appear less efficient. Further dialogue mixed with defensive feelings, may delay and complicate the feedback loop and potential improvements to the code. P6 mentioned: *"try to engage them... to understand what they mean by that review"* (P6), indicating delayed actions, compared to P2, whose emotional alignment seems to facilitate immediate actions to improve. Similar experience was echoed by P7, whose code was also reviewed by P11. He described her feedback as *"little bit harsh"*, yet willing to engage in a constructive discussion to resolve the comments, *"I would just have discussion with the reviewer ... we might come to a mutual agreement that would work for both of us ... I wouldn't just ignore"* (P7).

In comparison with an LLM-assisted review, the engineers in our sample show willingness to engage constructively with the LLM feedback, with a varying level of reservation, citing trust in LLMs, and lack of specific codebase context, as constraints. Despite the reservations, our interviewees seem to undertake a similar sense-making process of the feedback as for their peers. For example, while P18 has a clear preference for the LLM review, i.e., *"ChatGPT had done the best ... if I have to choose one of the four, including chatGPT, yes, I would choose chatGPT"* (P18), he still suggests combining peers' with LLM reviews to mitigate the lack of context. He explained his reasoning: *"AI might not be completely aware*

of what the context of the code you are writing ... what are your future goals, so how the code base should be" (P18).

Sense-making process: Sense-making is an active process of reflective evaluation of the feedback to pursue its adoption. Software engineers reflectively evaluate the feedback, then decide whether further dialogue with the reviewer is necessary before adoption. P5 summarizes this process: *"I went through each one line by line, or suggestion by suggestion ... and sort of evaluated it, and if I agree with it or not, some of them I did, some of them I didn't. For the ones that I did, I would have no problem making those changes. There were some good suggestions in there. Some of the ones that I didn't necessarily agree with or I had questions about ... I would open up a dialog with the other developer and sort of have a back and forth discussion with them about the suggestion"* (P5).

The reflective evaluation reconciles the external feedback with the software engineer's internal knowledge structures. P8 and P9 describe it as *"make sense to me"*, and *"does it make any sense"*, respectively. Then, if this sense-making is successful and perceived to contribute to improvements, then the engineers apply the feedback, e.g., *"If it does make any sense ... if I feel like they are more they're going to improve the readability and improve the style of coding. I'll go with that"* (P9).

Software engineers in our sample do not seem to discriminate in dealing with feedback, irrespective of the source, LLM or peers. They claim to undertake a similar sense-making process to evaluate and adopt LLM-generated feedback, despite its verbosity and high content, requiring more cognitive effort. For instance, P7 was asked about his decision-making process for ChatGPT-generate feedback, he replied, *"decision making, I still follow the same process I use with my peers"* (P7).

C. Reviewer context

Reviewer context are the inherent attributes or characteristics of the reviewer. For humans, it includes seniority and familiarity with the reviewer. While seniority exerts authority to adopt the feedback, familiarity reduces the emotional impact. P13 explains how he would process feedback from his peers: *"It depends who it's from. If it's from somebody who's more senior than me ... they know what they're talking about. If it was somebody not so senior, and maybe argue a bit"* (P13). This testimony shows that P13 attributes more credibility to his seniors' feedback compared to juniors.

Familiarity with peers fosters a sense of safety, reducing emotional tension because of the already existing relational capital. P15 explains: *"if the review comes from someone I know in real life, this kind of interaction also helps to digest the feedback and improve your personal relationship, whether good or bad review ... I know that person cares"* (P15).

For LLMs, trust in LLM abilities and lacking the codebase context in the review constrain engineers willingness to adopt LLM's review. P20 said: *"my experience, ChatGPT ... if you go a bit deeper in more advanced topics, it doesn't really know what to do clearly ... So I wouldn't trust ChatGPT"* (P20).

In this "human vs. machine" discovery, we learned that the introduction of LLMs brings a trade-off between cognitive and emotional engagement. Human reviewers have varied delivery styles, which may evoke stronger emotional responses, either positive or negative, influencing the process of acting upon the feedback. In contrast, LLM-assisted reviews, while more cognitively demanding, offer a consistent emotional experience. The resolution lies in engineers' personal skills and values to cope and their sense-making process, to decide, and adopt the feedback.

D. Human-AI collaboration

There was a clear divergence in preferences regarding how feedback should be delivered amongst the engineers in our sample. Some engineers preferred concise, actionable suggestions (e.g., P17, P18, and P20), while others wanted more detailed and educational feedback (e.g., P11 and P12). For example, P5 described his preference for LLM-generated feedback to have a high signal-to-noise ratio, containing a high proportion of useful, relevant, and actionable information (the "signal") relative to irrelevant or distracting content (the "noise"). He said: *"... there is a lot of information ... any point in here could be valid and worth considering, but I'm just like the signal to noise ratio, how much in here is like, truly valuable information that I would put a lot of thought into considering that's my concern"* (P5).

Willingness to adopt LLM in code review is marked by tension between the efficiency and technical completeness of the LLM-generated feedback and the desire for the human "touch" (P5) and expertise. When P12 was asked if she prefers P2 or ChatGPT reviews, she conveyed a strong preference: *"Well, ChatGPT being a faceless chat box is probably a factor. But what I've gathered from working with other people, especially people who are senior developers... they can give me more insights and things that I haven't previously seen and still connect them to the things that I'm currently working on. That I think is super valuable insight for me"* (P12). P15 echoes this tension in this statement: *"I can say ChatGPT one because it's more complete. So it found all the errors and all the things that I can improve. On the other hand, I can say I found the other participant's reviews more, let's say heartwarming, because some other person spent this time to read my code, understand it, and then do a review ... [ChatGPT] spent so one second"* (P15).

To balance LLM's efficiency with the human elements and expertise, most of our participants suggested combining both peers' and LLM's reviews. For example, even though P13 is willing to replace P16 and P17 with ChatGPT, he seems not willing yet to let go of the humans in the loop. *"I would replace two of them [i.e., P16 and P17] with ChatGPT, but I feel like if given the choice, I would use ChatGPT for a pre-review check. You could somehow get that to look over it while it's still in the IDE, make your changes, and then put it out for review and then let a person look at it"* (P13).

V. Discussion

In this section, we discuss the implications of our findings. Three main takeaways from this study: *Emotional intelligence (EI)*, *feedback constructiveness*, and *human-AI collaboration in SE*. The human review remains highly relevant, but the emotional impact is significant. EI has the potential to alleviate this emotional toll for an efficient process and the well-being of engineers. Our findings show diverse preferences for AI interaction, mainly the formulation of the feedback, as well as a willingness among engineers to adopt them. To capitalize on their capabilities, future AI tools should adapt to individual preferences.

A. Emotional intelligence

Our findings indicate that software engineers use different coping mechanisms to mitigate the emotional cost of engaging with peers' feedback. Some of these mechanisms are builtin and stem from their personal values (e.g., non-confrontational approach) or strategies (e.g., mindfulness) that they have developed. This implies the importance of fostering emotional intelligence amongst software engineers through training and support systems.

Humans are humans, and developers are humans, too! The expectation that an individual will perform consistently and constructively is perhaps unrealistic. Given the emotional variability inherent in human behaviors [45], fostering emotional intelligence is essential to help engineers navigate the psychological challenges posed when engaging with peer feedback [19], [45]. This built-in emotional intelligence has the potential to minimize frictions in the review process and shift the focus to code improvements, a primary objective of code review.

Significant work has been devoted to emotions in SE [63], [67]. Amongst others, topics range from emotions in requirements engineering, e.g., [18], [51], emotions in software artifacts, e.g., [61], developer's emotions, e.g., [27], [61], and the impact of emotions on software quality artifacts [39]. However, Sánchez-Gordón and Colomo-Palacios systematic review show that managing emotions is less explored [67], despite its potential to inform decision-making and social interactions [28].

Emotional intelligence (EI) is the individual ability to recognize, understand, and proactive manage emotions to navigate emotional situations in social interactions and subsequent decision-making [28].

Kosti et al. reported that emotional intelligence influences work preferences among software engineers [42]. They suggest that software engineers with higher emotional intelligence prefer being responsible for the entire development process and autonomy in prioritizing their tasks [42]. Rezvani and Khosravi found that EI mitigates stress and fosters trust among software developers, which results in increased performance [65].

Madampe et al. reported twelve EI strategies used by SE practitioners during requirements change handling [51]. These strategies have a direct impact on self-regulation of emotions and relationships. They range from peer support, openness in communication, and "social rituals" [51]. As our findings suggest, coping mechanisms are rooted in personal values like non-confrontational approaches or strategies like mindfulness, which help engineers to alleviate the emotional costs of feedback. These mechanisms can be viewed as manifestations of EI regulation [11].

Hodzic et al. suggest that EI "should be considered effective interventions" [32], [53]. This can be done through support programs within the organization or direct training [32], [53]. Therefore, we propose this implication:

Implication #1: Given its potential to minimize friction and contribute to more effective code reviews in the pursuit of improved code quality, EI training should become part of standard organizational support and development programs in SE.

B. Feedback constructiveness

Our findings suggest that constructive feedback requires lower cognitive demand and positive emotional responses. Gonçalves et al. report that negative feedback in code review causes interpersonal conflicts [79]. They suggest that developers should constructively manage their conflicts to foster a more positive code review environment [79]. Our findings show that the feedback delivery itself triggers negative emotional responses prior to further dialogue being initiated to resolve the comments. Hence, we suggest promoting feedback constructiveness to mitigate negative emotional responses and subsequent conflicts.

Kluger and DeNisi note that feedback content and delivery have varying effects on performance [40]. Constructive feedback is the most effective, as it focuses on the task and improvements, which minimizes negative emotional responses [40]. Hattie and Timperley suggest that feedback is constructive when it meets the criteria of being clear, task-oriented, and specific [31]. In educational settings, Carless suggests that feedback should aim to correct errors, foster a growth mindset, and encourage positive emotions to support learning [17]. Feedback should also be framed in a way that promotes cognitive engagement and reduces emotional defensiveness [71].

Implication #2: Feedback constructiveness should be part of computer science and software engineering educational programs. Organizations and SE teams should also document, promote, and reward constructive feedback.

C. Human-AI collaboration in SE

The transition from human-human to human-AI collaboration brings new complexities. Our findings show that while software engineers are generally willing to adopt LLM-assisted code review, there are varying preferences for how AI delivers feedback and an apparent tension between the efficiency and technical completeness of AI and the desire for the human "touch" and expertise.

The Sowa et al. study on human-AI collaboration in knowledge work suggests that when human-AI collaboration is enhanced, productivity increases [74]. They conclude that

AI automation should rather shift focus to collaborative approaches where humans and AI work closely together [74], balancing AI's computational strengths with the relational and experiential aspects of humans.

Studies in human-AI collaboration specific to traditional SE practices and processes remain scarce [30]. Hamza et al. study suggests that software engineers expect a "collaborative partner" not merely a tool. They further explain that this partnership is also expected to be effective and dialogic to ensure accurate and relevant AI output [30].

To ensure effective engagement, our findings indicate that willingness to adopt AI is characterized by varying preferences for interaction. In the context of LLM-assisted code review, while some software engineers prefer concise, actionable feedback that enables faster decision-making and implementation, others appreciate more detailed, educational feedback that contributes to their learning. We draw parallels with the Ghorbani et al. study on open-source contributors' preferences for GitHub bot interactions [26]. They found that developers prefer adaptable and customizable bot interactions, catering to individual preferences [26].

Jang et al. report that autistic worker preferred LLMs for their structured, clear responses and the privacy they offered, which is in contrast with the emotional risks associated with seeking help from humans [33]. AI can play a moderator role to make human feedback more constructive with a positive tone.

Implication #3: To capitalize on the potential of AI tools in SE processes, future AI tools should incorporate personalization features that align feedback delivery with individual preferences. In addition, AI can play the role of a moderator, adjusting or enhancing the tone and constructiveness of feedback to reduce the emotional cost and foster developer well-being.

Although our implications do not lead to concrete and actionable guidelines, they offer valuable directions for future research. For example, our implication for EI training does not specify exactly how training for EI is SE may be implemented. Future work could draw from the extensive work on psychology to develop tailored interventions and strategies for fostering emotional intelligence in software engineering contexts. Similarly, our implication on feedback constructiveness raises important questions about how to promote more effective and emotionally attuned feedback in a human-led review. Our findings on human-AI collaboration also open avenues for further research on what the future AI-assisted SE may look like and how to define this "AI partnership."

VI. Trustworthiness

To ensure our study's trustworthiness [55], we adopted these methods: *saturation*, *peer debriefing*, we provide a *thick description* (see Sect. III for these methods), and we conducted a *member checking* activity.

Member checking: We also carried out member checking [14] to seek feedback from our participants. This method allowed us to receive feedback on whether the analysis accurately represented our participants' experiences and perspectives, thereby adding a layer of verification to the accuracy of our interpretations [14]. We documented our preliminary findings in survey-like page forms and invited all participants to take part and comment on our interpretations, and 19 from 20 responded. Most comments were supportive of our interpretations and, in some instances, augmented our findings with additional data in the comments the participant provided. We paid an additional £5 for this effort. See Sect.III-E for supporting material and data.

VII. Limitations and Trade-offs

We conducted the code review process anonymously. We were constrained by the Prolific requirement to maintain participant anonymity throughout the study, whic is not the case in open source and proprietary software development. In the latter, developers are colleagues and interact directly on a daily basis. In such a social context, developers may align the tone of their feedback according to status, relationships, and team culture [13], [41]. Similarly, in open-source development, communities adopt different styles and strategies in pull request reviews, from "lenient" to "protective" according to their history, culture, and sustainability strategy [6], [7]. Aware of this limitation, we asked during the interview whether this factor has influenced our participants approach to reviews. Most of our participants claimed that was not the case.

The code reviews were conducted in an artificial setting characteristic of a research environment. Such a controlled environment may not capture the inherent complexity of a real-world setting. However, this controlled environment allowed us to focus on the variables of interest and gain nuanced findings [75]. We also acknowledge that our participants may have modified their approach to the review compared to a professional setting, given the research nature of their contribution. However, in the interview, we prompted all our participants to explain the content and the delivery style they used. This "reflective questioning" uncovered the "authentic" reasoning even when the original actions occurred in artificial settings [72].

VIII. Conclusion

In this study, we learned that in peer-led review, engagement is multi-dimensional, covering cognitive, emotional, and behavioral responses. Software engineers engage in a sense-making process before making a decision about the feedback suggestions. In an LLM-assisted review, the emotional toll is alleviated by the consistent positive delivery; however, the cognitive load to process the feedback is higher. LLM were perceived positively by software engineers for their technical superiority and completeness. However, their adoption seem to be contingent to diverse preferences on the feedback delivery.

Acknowledgment: We thank all the study participants for their time and effort. This study was funded by the department of computer science at Aalborg University; research funding for tenure-track assistant professors.

References

[1] "Github copilot · your ai pair programmer · github," https://github.com/features/copilot/, (Accessed on 08/27/2024).

[2] "Openai codex | openai," https://openai.com/index/openai-codex/, (Accessed on 08/27/2024).

[3] "Tabnine ai code assistant | private, personalized, protected," https://www.tabnine.com/, (Accessed on 08/27/2024).

[4] "Octoverse: The state of open source and rise of ai in 2023 - the github blog," https://github.blog/news-insights/research/the-state-of-open-source-and-ai/, 2023, (Accessed on 08/27/2024).

[5] E. Al Haque, C. Brown, T. D. LaToza, and B. Johnson, "Information seeking using AI assistants," 2024. [Online]. Available: https://arxiv.org/abs/2408.04032

[6] A. Alami, M. L. Cohn, and A. Wąisowski, "How do foss communities decide to accept pull requests?" in *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, 2020, pp. 220–229.

[7] A. Alami, R. Pardo, M. L. Cohn, and A. Wąsowski, "Pull request governance in open source communities," *IEEE Transactions on Software Engineering*, vol. 48, no. 12, pp. 4838–4856, 2021.

[8] A. Alami, M. Zahedi, and N. Ernst, "Are you a real software engineer? best practices in online recruitment for software engineering studies," in *Proceedings of the 1st IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering*, 2024, pp. 52–57.

[9] K. M. Aldiabat and C.-L. Le Navenec, "Data saturation: The mysterious step in grounded theory methodology," *The qualitative report*, vol. 23, no. 1, pp. 245–261, 2018.

[10] C. Anthony, B. A. Bechky, and A.-L. Fayard, ""collaborating" with ai: Taking a system view to explore the future of work," *Organization Science*, vol. 34, no. 5, pp. 1672–1694, 2023.

[11] R. Bar-On, "The bar-on model of emotional-social intelligence (esi) 1," *Psicothema*, pp. 13–25, 2006.

[12] G. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interacting with computers*, vol. 23, no. 1, pp. 4–17, 2011.

[13] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *Proceedings of the 2006 international workshop on Mining software repositories*, 2006, pp. 137–143.

[14] L. Birt, S. Scott, D. Cavers, C. Campbell, and F. Walter, "Member checking: a tool to enhance trustworthiness or merely a nod to validation?" *Qualitative health research*, vol. 26, no. 13, pp. 1802–1811, 2016.

[15] G. A. Bowen, "Naturalistic inquiry and the saturation concept: a research note," *Qualitative research*, vol. 8, no. 1, pp. 137–152, 2008.

[16] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[17] D. Carless, "Differing perceptions in the feedback process," *Studies in higher education*, vol. 31, no. 2, pp. 219–233, 2006.

[18] R. Colomo-Palacios, A. Hernández-López, Á. García-Crespo, and P. Soto-Acosta, "A study of emotions in requirements engineering," in *Organizational, Business, and Technological Aspects of the Knowledge Society: Third World Summit on the Knowledge Society, WSKS 2010, Corfu, Greece, September 22-24, 2010. Proceedings, Part II 3*. Springer, 2010, pp. 1–7.

[19] K. M. Connor and J. R. Davidson, "Development of a new resilience scale: The connor-davidson resilience scale (cd-risc)," *Depression and anxiety*, vol. 18, no. 2, pp. 76–82, 2003.

[20] J. W. Creswell and C. N. Poth, *Qualitative inquiry and research design: Choosing among five approaches*. Sage publications, 2016.

[21] E. Dehaerne, B. Dey, S. Halder, S. De Gendt, and W. Meert, "Code generation using machine learning: A systematic review," *IEEE Access*, vol. 10, pp. 82 434–82 455, 2022.

[22] J. Del Sagrado, I. M. Del Águila, and F. J. Orellana, "Multi-objective ant colony optimization for requirements selection," *Empirical Software Engineering*, vol. 20, pp. 577–610, 2015.

[23] P. Denny, J. Prather, B. A. Becker, C. Mooney, J. Homer, Z. C. Albrecht, and G. B. Powell, "On designing programming error messages for novices: Readability and its constituent factors," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21, May 2021, p. 1–15.

[24] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," in *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE, 2023, pp. 31–53.

[25] J. A. Fredricks, P. C. Blumenfeld, and A. H. Paris, "School engagement: Potential of the concept, state of the evidence," *Review of educational research*, vol. 74, no. 1, pp. 59–109, 2004.

[26] A. Ghorbani, N. Cassee, D. Robinson, A. Alami, N. A. Ernst, A. Serebrenik, and A. Wąsowski, "Autonomy is an acquired taste: Exploring developer preferences for github bots," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1405–1417.

[27] D. Girardi, F. Lanubile, N. Novielli, and A. Serebrenik, "Emotions and perceived productivity of software developers at the workplace," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3326–3341, 2021.

[28] D. Goleman, *Emotional intelligence: Why it can matter more than IQ*. Bloomsbury Publishing, 2020.

[29] J. J. Gross, "The emerging field of emotion regulation: An integrative review," *Review of general psychology*, vol. 2, no. 3, pp. 271–299, 1998.

[30] M. Hamza, D. Siemon, M. A. Akbar, and T. Rahman, "Human-ai collaboration in software engineering: Lessons learned from a hands-on workshop," in *Proceedings of the 7th ACM/IEEE International Workshop on Software-intensive Business*, 2024, pp. 7–14.

[31] J. Hattie and H. Timperley, "The power of feedback," *Review of educational research*, vol. 77, no. 1, pp. 81–112, 2007.

[32] S. Hodzic, J. Scharfen, P. Ripoll, H. Holling, and F. Zenasni, "How efficient are emotional intelligence trainings: A meta-analysis," *Emotion Review*, vol. 10, no. 2, pp. 138–148, 2018.

[33] J. Jang, S. Moharana, P. Carrington, and A. Begel, ""it's the only thing i can trust": Envisioning large language model use by autistic workers for communication assistance," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–18.

[34] W. Jiao and H. Mei, "Automated adaptations to dynamic software architectures by using autonomous agents," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 7, pp. 749–770, 2004.

[35] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?" in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. IEEE Press, 2013, p. 672–681.

[36] C. Jw, "Qualitative inquiry and research design," *Choosing Among Five Traditions*, 1998.

[37] E. R. Kahu, "Framing student engagement in higher education," *Studies in higher education*, vol. 38, no. 5, pp. 758–773, 2013.

[38] E. Kalliamvakou, C. Bird, T. Zimmermann, A. Begel, R. DeLine, and D. M. German, "What makes a great manager of software engineers?" *IEEE Transactions on Software Engineering*, vol. 45, no. 1, pp. 87–106, 2017.

[39] K. M. Khan and M. Saleh, "Understanding the impact of emotions on the quality of software artifacts," *IEEE Access*, vol. 9, pp. 110 194–110 208, 2021.

[40] A. N. Kluger and A. DeNisi, "The effects of feedback interventions on performance: a historical review, a meta-analysis, and a preliminary feedback intervention theory." *Psychological bulletin*, vol. 119, no. 2, p. 254, 1996.

[41] O. Kononenko, O. Baysal, and M. W. Godfrey, "Code review quality: How developers see it," in *Proceedings of the 38th international conference on software engineering*, 2016, pp. 1028–1038.

[42] M. V. Kosti, R. Feldt, and L. Angelis, "Personality, emotional intelligence and work preferences in software engineering: An empirical study," *Information and Software Technology*, vol. 56, no. 8, pp. 973–990, 2014.

[43] S. Kvale and S. Brinkmann, *Interviews: Learning the craft of qualitative research interviewing*. sage, 2009.

[44] M. Larkin, P. Flowers, and J. A. Smith, *Interpretative phenomenological analysis: Theory, method and research*. sAgE Publications ltd, 2021.

[45] R. S. Lazarus, *Stress, appraisal, and coping*. Springer, 1984, vol. 464.

[46] Z. Li, S. Lu, D. Guo, N. Duan, S. Jannu, G. Jenks, D. Majumder, J. Green, A. Svyatkovskiy, S. Fu, and N. Sundaresan, "Automating code review activities by large-scale pre-training," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE '22, Nov. 2022.

[47] J. T. Liang, C. Yang, and B. A. Myers, "A large-scale survey on the usability of ai programming assistants: Successes and challenges,"

in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. ACM, Feb. 2024, p. 1–13.

[48] W. Ling, P. Blunsom, E. Grefenstette, K. M. Hermann, T. Kočiský, F. Wang, and A. Senior, "Latent predictor networks for code generation," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 599–609. [Online]. Available: https://aclanthology.org/P16-1057

[49] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[50] J. Lu, L. Yu, X. Li, L. Yang, and C. Zuo, "Llama-reviewer: Advancing code review automation with large language models through parameter-efficient fine-tuning," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, Oct. 2023, p. 647–658.

[51] K. Madampe, R. Hoda, and J. Grundy, "A framework for emotion-oriented requirements change handling in agile software engineering," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3325–3343, 2023.

[52] T. W. Malone, D. Rus, and R. Laubacher, "Artificial intelligence and the future of work," *A report prepared by MIT Task Force on the work of the future, Research Brief*, vol. 17, pp. 1–39, 2020.

[53] V. Mattingly and K. Kraiger, "Can emotional intelligence be trained? a meta-analytical investigation," *Human Resource Management Review*, vol. 29, no. 2, pp. 140–155, 2019.

[54] L. Mich and R. Garigliano, "Nl-oops: A requirements analysis tool based on natural language processing," *WIT Transactions on Information and Communication Technologies*, vol. 28, 2002.

[55] M. B. Miles, A. M. Huberman, and J. Saldaña, *Qualitative data analysis: A methods sourcebook. 3rd*. Thousand Oaks, CA: Sage, 2014.

[56] M. B. Miles, M. Huberman, and J. Saldana, *Qualitative data analysis: A methods sourcebook*. SAGE Publications, Incorporated, 2013.

[57] M. Monperrus, "Explainable software bot contributions: Case study of automated bug fixes," in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, May 2019.

[58] M. Monperrus, S. Urli, T. Durieux, M. Martinez, B. Baudry, and L. Seinturier, "Repairnator patches programs automatically," *Ubiquity*, vol. 2019, no. July, p. 1–12, Jul. 2019.

[59] J. M. Morse, "Theoretical saturation," *Encyclopedia of social science research methods*, vol. 3, pp. 1122–3, 2004.

[60] A. Murgia, D. Janssens, S. Demeyer, and B. Vasilescu, "Among the machines: Human-bot interaction on social q&a websites," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI'16. ACM, May 2016. [Online]. Available: http://dx.doi.org/10.1145/2851581.2892311

[61] A. Murgia, P. Tourani, B. Adams, and M. Ortu, "Do developers feel emotions? an exploratory analysis of emotions in software artifacts," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 262–271.

[62] G. Ninaus, A. Felfernig, M. Stettinger, S. Reiterer, G. Leitner, L. Weninger, and W. Schanil, "Intellireq: intelligent techniques for software requirements engineering," in *ECAI 2014*. IOS Press, 2014, pp. 1161–1166.

[63] N. Novielli and A. Serebrenik, "Sentiment and emotion in software engineering," *IEEE Software*, vol. 36, no. 5, pp. 6–23, 2019.

[64] S. Panichella and N. Zaugg, "An empirical investigation of relevant changes and automation needs in modern code review," *Empirical Software Engineering*, vol. 25, no. 6, p. 4833–4872, Sep. 2020. [Online]. Available: http://dx.doi.org/10.1007/s10664-020-09870-3

[65] A. Rezvani and P. Khosravi, "Emotional intelligence: The key to mitigating stress and fostering trust among software developers working on information system projects," *International Journal of Information Management*, vol. 48, pp. 139–150, 2019.

[66] G. Rodríguez, Á. Soria, and M. Campo, "Artificial intelligence in service-oriented software design," *Engineering Applications of Artificial Intelligence*, vol. 53, pp. 86–104, 2016.

[67] M. Sánchez-Gordón and R. Colomo-Palacios, "Taking the emotional pulse of software engineering—a systematic literature review of empirical studies," *Information and Software Technology*, vol. 115, pp. 23–43, 2019.

[68] K. Schneid, L. Stapper, S. Thöne, and H. Kuchen, "Automated regression tests: A no-code approach for bpmn-based process-driven applications," in *2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC)*, 2021, pp. 31–40.

[69] S. H. Schwartz, "Universals in the content and structure of values: Theoretical advances and empirical tests in 20 countries," *Advances in experimental social psychology/Academic Press*, 1992.

[70] I. Seidman, *Interviewing as qualitative research: A guide for researchers in education and the social sciences*. Teachers college press, 2006.

[71] V. J. Shute, "Focus on formative feedback," *Review of educational research*, vol. 78, no. 1, pp. 153–189, 2008.

[72] D. Silverman, "What counts as qualitative research? some cautionary comments," *Qualitative sociology review*, vol. 9, no. 2, pp. 48–55, 2013.

[73] E. A. Skinner, T. A. Kindermann, and C. J. Furrer, "A motivational perspective on engagement and disaffection: Conceptualization and assessment of children's behavioral and emotional participation in academic activities in the classroom," *Educational and psychological measurement*, vol. 69, no. 3, pp. 493–525, 2009.

[74] K. Sowa, A. Przegalinska, and L. Ciechanowski, "Cobots in knowledge work: Human–ai collaboration in managerial professions," *Journal of Business Research*, vol. 125, pp. 135–142, 2021.

[75] K.-J. Stol and B. Fitzgerald, "The abc of software engineering research," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 3, pp. 1–51, 2018.

[76] A. Svyatkovskiy, S. K. Deng, S. Fu, and N. Sundaresan, "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*, 2020, pp. 1433–1443.

[77] R. Tufano, S. Masiero, A. Mastropaolo, L. Pascarella, D. Poshyvanyk, and G. Bavota, "Using pre-trained models to boost code review automation," in *Proceedings of the 44th International Conference on Software Engineering*. ACM, May 2022.

[78] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[79] P. Wurzel Goncalves, J. SV Goncalves, and A. Bacchelli, "Constructive code review: Managing the impact of interpersonal conflicts in practice," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, 2024, pp. 334–345.

[80] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," in *2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*. Association for Computational Linguistics, 2018, pp. 3911–3921.

[81] R. N. Zaeem, M. R. Prasad, and S. Khurshid, "Automated generation of oracles for testing user-interaction features of mobile apps," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*. IEEE, 2014, pp. 183–192.

[82] L. M. A. Zohair, "The future of software engineering by 2050s: Will ai replace software engineers?" *International Journal of Information Technology*, vol. 2, no. 3, pp. 1–13, 2018.