# Human-Machine Teaming and Team Effectiveness in AI tools for Software Engineering

Irum Rauf
*The Open University, UK*
*Lero the Research Ireland Centre for Software*
irum.rauf@open.ac.uk

Helen Sharp
Tamara Lopez, Michel Wermelinger
*The Open University, UK*
firstname.lastname@open.ac.uk

***Abstract*—*Background:*** **Artificial Intelligence (AI) is increasingly being used to support software engineering (SE), shifting the role of AI tools for SE (AI4SE) towards team members rather than simply tools. Human-machine teaming (HMT) views humans and machines as members of a team. This approach may be useful in AI4SE but HMT and team effectiveness have received little attention in the software engineering community. *Objective:* To unpick whether and how HMT and team effectiveness have been applied in AI4SE, and to identify recommendations. *Method:* A mapping study approach identifies 21 papers on developer-centred AI4SE. Each one is analysed to identify its use of HMT, its focus regarding AI-developer interaction, and any elements of team effectiveness. *Results:* Different aspects of AI-developer interaction have been considered but only one paper explicitly uses the HMT concept. Some elements of team effectiveness are considered, but others are not addressed at all, and several studies do not consider any. *Conclusion:* Researchers are encouraged to consider HMT as a framework for AI4SE, to leverage research in all-human teams, adaptive UX and developer motivation to inform tool design, and reduce reliance on human monitoring.**

## I. Introduction

Artificial Intelligence (AI) is increasingly being used to solve software engineering (SE) challenges (referred to as AI4SE [1]), expanding its role beyond a tool that just automates tasks, towards being an autonomous decision-maker independent of human control. In response, the role of human developers needs to evolve from being simply a user [2], so that developers and AI tools work together as team members. Effective teams are valuable assets and the notion of *team* is being expanded to include human-machine partnerships [3]. Human-machine teaming (HMT) promotes effective teamwork and aims to foster collaborations between machines and humans, emphasizing interaction and partnership, while capitalizing on the unique strengths of them both [4] [2].

The field of HMT is relatively well-known in areas such as defence [5] and aviation [6]. However, the concept of HMT is less explored in SE. Lo [7] suggests that the effective collaboration between AI tools and practitioners is hindered due to a mismatch between the levels of abstraction at which they operate: AI tools typically handle low-level tasks like code generation and fault localization, while practitioners work at higher levels of abstraction with broader objectives. Furthermore, the psychological experiences of humans interacting with autonomous agents can vary and may influence the outcomes of future collaborations [8].

Building on these ideas, this paper seeks to unpick whether and how the concept of HMT, i.e. an effective AI-developer team, has been used in AI4SE. This is done using a mapping study approach [9] to identify 21 developer-centred AI4SE studies. 'Developer-centred' studies focus on how developers and AI tools interact and how this influences their workflows and practices. Each study was analysed first to identify its focus in terms of AI-developer interaction, and whether it uses the HMT concept. The papers were then analysed deductively using Salas et al's [10] model of team effectiveness. The findings show that HMT is not widely used and many studies focus on the user experience of AI tools, while some aspects of team effectiveness are considered, but others are not.

## II. Related Work

### A. Artificial Intelligence for Software Engineering (AI4SE)

How SE and AI may be combined has been explored for many years [11] but the field of AI4SE has experienced significant recent growth, leading to several systematic reviews.

Feldt et al [12] review papers from the RAISE workshop (Realizing Artificial Intelligence Synergies in Software Engineering). They present a taxonomy for AI in SE Application Levels (AI-SEAL), which categorizes AI applications based on their point of application, the type of AI used, and the level of automation, with a focus on human involvement in decision-making. This provides a framework for examining the use of AI in SE through a structured, application-level perspective. Sofian et al [13] provide an overview of AI techniques applied across different stages of SE, primarily positioning humans as users of AI tools. Their systematic mapping study, covering literature from 2015 to 2021, reveals that AI methods are most commonly used in the requirements engineering phase, with machine learning (ML) being the predominant technique.

Other systematic literature reviews examine the use of specific AI approaches in SE. For instance, Watson et al [14] and Yang et al [15] explore the application of deep learning in SE research, while Mohammadkhani et al [16] investigate explainable AI4SE and Hou et al [17] focus on large language models. These highlight different features of AI4SE but focus on technical rather than teaming aspects of AI-developer interaction.

### B. Human-Machine Teaming (HMT)

HMT envisions humans collaborating with intelligent machines that can understand human capabilities as team members [2]. In this dynamic relationship, humans need to trust the machines and have transparent insight into their decisions, allowing for joint action towards achieving shared goals [18].

The importance of integrating the concepts and characteristics of teaming in HMT are well-understood. McNeese et al [19] show the potential of HMT to overcome shortcomings of all-human teams. They looked at team situation awareness (TSA) and team conflict in HMT through a controlled experiment and found that TSA improved over time compared to all-human teams, and that team conflicts negatively impacted TSA in all-human teams. Walliser et al [20] demonstrated that team-building interventions in HMT with autonomous agents significantly improved various teamwork outcomes, including performance. Johnson et al [21] emphasize the need to understand interdependence in HMT instead of focusing on task work only and they offer a tool that helps recognize both human and technological factors that enhance or inhibit effective teaming. In contrast to these works that focus on machines, Dubrow and Orvis [22] investigate humans who can be effective human-machine teammates, e.g. those open to new experiences, have tolerance for ambiguity, and high propensity to trust. Moreover, their work highlights the importance of training individuals on how machines make decisions.

### C. Team Effectiveness

Effective teamwork in SE is essential, whether HMT or all-human. Effective teamwork in all-human software teams is challenging, including issues with communication, learning, prioritization, and leadership [23]; team performance is often measured by achieving project goals, staying within budget, and delivering high-quality software [24]. Several team effectiveness models have been developed [25], including Salas et al's [10] Big Five Model of Team Effectiveness (Fig 1).

This model has five key components that contribute to successful teamwork. *Team leadership* is directing and coordinating the activities of other team members. *Team orientation* refers to the degree to which team members have a collective attitude towards collaboration and working together. *Mutual performance monitoring* is the ability to accurately monitor teammate performance. *Adaptability* is the ability to adjust strategies or actions in response to changing conditions. *Backup behaviour* is the ability to anticipate team members' needs and shift workload. Three coordinating mechanisms facilitate these components: *shared mental models* (an understanding of relationships between tasks and how team members interact), *closed-loop communication* refers to the exchange of information between a sender and a receiver, regardless of the medium, and *mutual trust* (a shared belief that team members do their best).

## III. The Research Approach

Our goal is to gain insights about the use of HMT and team effectiveness concepts in AI4SE by analysing existing

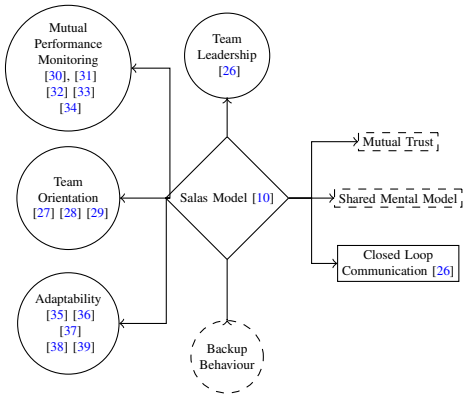


Fig. 1. Salas' Big Five Model of Team Effectiveness. Circles = components, rectangles = coordination mechanisms, dashed edges = no relevant studies

developer-centred studies. Our research questions are: (a) Has the concept of HMT been used in developer-centred AI4SE studies? (b) What aspects of AI-developer interaction have been investigated in these studies? (c) What team effectiveness elements have been considered in these studies?

The terms human-machine teaming, human-autonomy teaming, human-agent teaming and human-machine collaboration are often used interchangeably. Hence, our search string was:

```
{( human AND (machine OR autonomy OR agent OR AI)
   AND (teaming OR collaboration ) AND/OR
   software OR (development OR engineering))
< in abstract, keywords, and title > }
```

Three inclusion criteria were used: explicitly addressing HMT in AI4SE; empirical studies; studies including an autonomous agent working alongside humans. Two exclusion criteria were: papers exploring AI software tools that automate specific tasks; thought pieces without empirical work.

The review targeted: *International Conference on Software Engineering (ICSE)*, *International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, *ACM Transactions on Software Engineering and Methodology (TOSEM)* and *Empirical Software Engineering Journal (ESE)*, using the ACM Digital Library for ICSE and CHASE, and TOSEM and ESE online repositories. We focused on these because they are top-tier venues that feature multidisciplinary research, especially developer studies, which aligns well with our aim. Future studies will consider a broader set of sources, but this initial scope ensures feasibility and relevance.

The first author filtered the search results using the inclusion and exclusion criteria by examining titles and keywords, and then reviewed the abstracts of the remaining articles. This resulted in 14 articles, which were reviewed in full. We identified seven new studies through *forward snowballing* and *backward snowballing* [40]; reading their abstracts and removing duplicates left five new studies. Through Google Scholar, which indexes several databases [41], we found seven more studies, two of which were included. This resulted in 21 studies. The second author reviewed these articles and confirmed their relevance.

The first author analysed each paper inductively to identify

its use of the HMT concept, and its focus in terms of AI-developer interaction, and then performed a deductive analysis using the eight elements of Salas' model [10]. These analyses were discussed iteratively with the second author. All authors discussed the overall findings and recommendations.

Due to the studies' diverse empirical methods and techniques, we used narrative synthesis to gather insights [42]. In this, findings from multiple studies are brought together using words and text to summarize and compare the results [43].

## IV. Findings

Table 1 summarises the 21 papers. It shows increased activity recently, which coincides with the release of GitHub CoPilot in 2021 and ChatGPT in 2022. This section presents narrative synthesis highlights for each research question.

### *A. Has the concept of HMT been used in developer-centred AI4SE studies?*

Only one paper [26] explicitly uses the term "HMT" in the context of SE. However, many other studies implicitly address elements critical to HMT, such as collaboration between developers and AI tools, and how adaptability occurs when humans and AI work together. For more discussion, see Section IV-B.

### *B. What aspects of AI-developer interaction have been investigated in developer-centred AI4SE studies?*

*1) Social Interaction and Collaboration (one study):* Zanetti et al [27] highlight the importance of *social interaction* within developer *collaboration* networks, and how studying this can predict efficient, practical methods to identify valid bug reports. They calculated the centrality of bug reporters in the network, analysed bug reports and used quantitative metrics to predict which of them are likely to be valid.

*2) Subjectivity and Perceptions (five studies):* Hozano et al [30] take account of developers' *subjective* view of code smells through an investigation of machine learning (ML) algorithm accuracy in detecting code smells for different developers. The ML algorithms did not reach high accuracy. Ziegler et al [29] investigate how Copilot users *think* it affects their productivity. They found that how often developers accepted Copilot's suggestions was more strongly correlated with reported productivity than persistence. De Souza et al [48] investigate how AI code assistants are *perceived* by developers based on Twitter discussions. They found that developers value their productivity and versatility but have concerns about code quality, security and ethics. Zheng [47] built a predictive model of *developers' attitudes* towards the future of AI and identify potential predictors, such as years of coding. Melegati et al [49] explore developers' *perceptions* of AI-based tools using a fictional futuristic scenario. They found concerns about lack of trust, loss of control, fear of being replaced by AI, but positive attitudes towards AI's potential benefits.

*3) Knowledge Sharing and Learning in Online Communities (two studies):* Bangash et al [50] analyse posts on Stack Overflow (SO) to show *how knowledge is shared* in this community. They identify developers' educational needs and how *online communities* can be more helpful. Newton et al. [26] examine the link between the usage of bots that help and direct team activities, and productivity outcomes. They show how bots can facilitate the *exchange of information* within software development teams and support team members to learn from each other and maintain awareness, leading to more efficient *knowledge sharing* compared to all-human teams.

*4) Non-functional aspects (4 studies, 3 on Copilot and security):* Asare et al [34] and Pearce et al [31] analyse AI-generated code from a *security* perspective. Asare et al [34] compared *vulnerabilities* in code generated by Copilot and by developers. They found that Copilot avoids security flaws, and does not introduce vulnerabilities more frequently or severely than humans. Pearce et al [31] examine the circumstances under which Copilot might create insecure code. They suggest that developers use robust validation practices and a critical mindset to assess AI-generated code for *security* and *integrity*. Klemmer et al [39] found that despite *security* and *quality* concerns, participants widely use AI assistants for security tasks and vulnerability detection.

Nascimento et al [38] asked whether a developer could solve a task better than an automated system. They compared the *performance* and *memory efficiency* of programs written by humans and ChatGPT. The findings suggest that ChatGPT enhances performance for some problems compared to novices but not experts, and it enhances memory efficiency in some problems compared to both experts and novices.

*5) User Experience (nine studies, five on Copilot):* Jiang et al [35] investigated the *user experience* of natural language code synthesis with a large generative language model. They found challenges with its use, including the need to learn the model's syntax, and difficulty in forming an accurate mental model of requests the model could handle. Xu [45] *evaluated* how using machine learning methods for code generation and retrieval from natural language queries affects developer *experience*. Mendes et al [46] examine engineers' *experiences* using AI code assistants in practice. They find faster coding, more focus on complex tasks, improved code quality and a helpful start on unfamiliar tasks, but low code accuracy and disruptive suggestions that hinder developers' comfort.

Studies on Copilot had mixed results. Imai [32] investigates the *effectiveness of using* Copilot for pair programming compared to human-only pairs. They show that while Copilot increases the number of lines of code added, it often produces lower-quality code. Bird et al [28] focus on capturing *experience* of developers pairing with Copilot. The results indicated that pairing with Copilot encourages creativity, but risks exposing sensitive information like API keys and passwords. Nguyen and Nadi [44] assess Copilot's code across multiple languages and found that it could be simplified and sometimes used undefined methods, causing concerns of *understandability*. Vaithilingam et al [36] found that participants had difficulty *understanding, editing*, and *debugging* Copilot code snippets, which hindered their effectiveness. Barke et al [37] identified two *interaction modes* when using Copilot: in 'acceleration' mode, the developer understands their next steps, and Copilot

TABLE 1
EMPIRICAL STUDIES OF DEVELOPER-CENTRED RESEARCH ON USE OF AI-BASED TOOLS IN SOFTWARE ENGINEERING

| AI-developer interaction | Team effectiveness element | Citation | What did they do? |
|---|---|---|---|
| User Experience | Team Orientation | Bird [28] | Forum discussion analysis, case study, and large-scale survey on Copilot |
| | Adaptability | Jiang [35] | 14-participant user study on impact of LLMs on software development |
| | | Vaithilingham [36] | 24-participant user study on how developers use and perceive Copilot |
| | | Barke [37] | 20-participant user study using Copilot for programming tasks |
| | Mutual Performance | Imai [32] | 21-student experiment of pair programming with Copilot |
| | | Dakhel [33] | Evaluations with fundamental algorithmic problems and solutions |
| | None | Nguyen&Nadi [44] | Evaluated Copilot's code in multiple languages for 33 LeetCode queries |
| | | Xu [45] | 31-participant controlled study on the use of NL2Code assistance. |
| | | Mendes [46] | 14 interviews with software developers |
| Nonfunctional Aspects | Mutual Performance | Pearce [31] | Analysed 1,689 programs poduced by Copilot for security weaknesses |
| | | Asare [34] | Compared Copilot C/C++ vulnerability suggestions to developers' code |
| | Adaptability | Nascimento [38] | Assessed ChatGPT code versus developer code uploaded in Leetcode |
| | | Klemmer [39] | 27 interviews with developers, review of 190 Reddit posts & comments |
| Subjectivity & Perceptions | Team Orientation | Ziegler [29] | Usage metrics and Survey |
| | Mutual Performance | Hozano [30] | Analyzed performance of 6 widely used algorithms detecting 4 types of code smells, based on data from 40 diverse developers |
| | None | Zheng [47] | Neural network & logistic regression of SO 2018 Developer Survey |
| | | deSouza [48] | Quantitative & qualitative analysis of 331 Copilot and Tabnine tweets |
| | | Melegati [49] | Social science fiction study with 38 students |
| Knowledge Sharing & Learning | Team Leadershp, Closed-loop Communication | Newton [26] | Statistical analysis of open source project data extracted from GitHub |
| | None | Bangash [50] | Used topic modeling approach to analyse ML related posts on SO |
| Social Interaction & Collaboration | Team Orientation | Zanetti [27] | Social network analysis of 700,000 bug reports obtained from Mozilla Firefox, Mozilla Thunderbird, Eclipse and NetBeans |

expedites the process; in 'exploration' mode, the developer is uncertain and relies on Copilot to explore options or find a starting point. Dakhel et al [33] sought to *understand how developers can benefit* from Copilot effectively. They found that novices struggle to identify and filter out buggy solutions.

### C. What team effectiveness elements have been considered in developer-centered AI4SE studies?

Figure 1 highlights which studies address specific elements of Salas et al's [10] model.

*1) Team Leadership (one study):* Newton et al [26]'s study indicates that including bots that help and direct team activities increases productivity, work centralisation and communication and helps plan work more effectively than human-only teams. These findings resonate with key aspects of team leadership, such as motivating, directing and coordinating team members.

*2) Team Orientation (three studies):* These studies provide insight into humans and AI tools working together towards shared goals, a key indicator of team orientation. Zanetti et al [27] focus on efficient bug triaging as a shared goal. By leveraging the social dynamics of collaboration networks, they emphasize how human developers and tools can foster synergy and collaboration to achieve this goal. Bird et al [28] extend the discussion by examining AI pair programming tools. Their findings reveal that developers spend more time reviewing code than writing it, underscoring the need for developers to acquire skills in code review. This shift reflects a team-oriented mindset, where developers prioritize collaboration, adapt to others' needs, and embrace shared responsibility which are key for fostering team effectiveness. Similarly, Ziegler et al [29] found that the acceptance rate of Copilot's code suggestions correlated more strongly with reported productivity than persistence. This highlights the collaborative value of Copilot in facilitating progress toward team goals.

*3) Mutual Performance Monitoring (five studies):* Mutual performance monitoring between human teammates is bidirectional, but four of these five studies focus only on developers monitoring AI output rather than AI monitoring human performance. Hozano et al [30] is the only study examining how AI tools align with developers' perceptions. Pearce et al [31] and Imai [32] focus on developers monitoring Copilot. Pearce et al [31] used a pair programming setting and found that, as with humans, Copilot can generate vulnerable code so developers need to critically assess its outputs. Imai [32] also emphasize that Copilot can enhance productivity but requires human oversight to maintain code quality. Dakhel et al [33] suggest that a developer's coding experience will influence the quality of code produced. Asare et al [34] compare code generated by Copilot and by humans, assessing how it performs as a "team member" and its contributions to the coding process. These studies underscore the importance of combining the strengths of both parties to achieve optimal outcomes.

*4) Adaptability (five studies):* These studies focus on some form of adjustment made for developers and AI tools to work together. Jiang et al [35] found that developers had to adjust their mental model of what could be reliably interpreted by a code synthesis tool. Vaithilingam et al [36] identified two main strategies to adapt to Copilot's limitations: fixing incorrect code or discontinuing its use. Klemmer et al [39] observed

that developers distrust AI-generated code and need to review it. Barke et al [37] recommend that AI assistants adapt better to developers' needs depending on the mode. For instance, if the developer is in acceleration mode, it could offer concise, high-confidence code that minimizes disruptions. Nascimento et al [38] emphasize the importance of adaptability in task allocation, suggesting that developers' expertise and the specific task at hand should influence how AI tools interact with them.

*5) Closed-loop Communication (one study):* Only one study considered communication between team members [26]. Newton et al investigate the role of software bots that prompt users to complete tasks and provide requested information. Their findings suggest that, in facilitating communication and collaboration within hybrid human-bot teams, such bots can enhance team task output and actively foster collaboration.

*6) Lack of consideration (seven studies):* Five studies do not consider team effectiveness at all [44] [45] [47] [48] [50]. Two studies acknowledge the influence of AI4SE but do not explore its effect on team dynamics. Melegati et al [49] argue that it is premature to assess team-level implications of future AI tools. Mendes et al [46] highlight that AI4SE is initiating a transformative shift in software development practices but stop short of examining its effects on team performance.

## V. Discussion

This mapping study aimed to uncover whether HMT has been used in AI4SE, and what aspects of AI-developer interaction and team effectiveness have been addressed in developer-centred empirical studies. Despite a recent increase in research efforts, we found only one study explicitly using HMT to investigate autonomous agents in SE [26]. Nine of the 21 studies concentrated on developers' user experience. Although this is important for effective development, it does not, on its own, support the concept of HMT. Other aspects of AI-developer interaction such as developer perceptions, knowledge sharing and collaboration are part of teamwork but they were not placed in an overarching framework such as HMT. Applying HMT results in this way will have its challenges. HMT takes a holistic team-level view of interaction. To extend AI4SE tools to team-level concerns may require better training data than is currently available. For example, understanding architecture and design to answer queries, make recommendations and share knowledge among the team would require a lot of good training material for a general ML model to learn the architecture of a large codebase. In addition, AI is not currently helpful for legacy code, e.g. Fortran, because of a lack of good training data. Even with better data, more sophisticated AI technology may also be needed.

Some studies touch upon Salas' elements of team effectiveness but they do not focus on them, nor do they explicitly refer to them in the context of HMT. Importantly, although several studies are categorized under 'mutual performance', these studies primarily offer only one-way performance monitoring (machine-human or human-machine). Moreover, three of the elements were not addressed in any study: backup behaviour, shared mental models, and mutual trust.

These findings have limitations due to the sources chosen and potential researcher bias but some initial implications are:

*1) Consider HMT as a guiding framework:* All of the identified studies consider aspects that could help create a hybrid AI-human SE team, but they aren't treated holistically. The HMT concept offers a perspective on human-machine interaction that focuses on forming a *team*. Effective HMT in AI4SE could encourage trust in software teams which aligns with responsible SE initiatives [51].

*2) Draw on all-human team research:* HMT may have different characteristics than all-human teams, but research from organisational psychology and other disciplines may provide insights for evaluating and designing AI4SE. This work used Salas' effectiveness model but there are others [24].

*3) Leverage previous research in adaptive UX:* Good UX is important but AI tools will need to adapt to their human collaborators. Adaptive UX provides some useful techniques for adaptive AI tools [52].

*4) Moderate reliance on human monitoring:* Many studies found that AI tools require human monitoring. Careful monitoring is a key element of team resilience [53], but humans are not good at monitoring and practitioners are instead motivated by problem solving and creativity [54].

## VI. Conclusion

The landscape of research in AI4SE is evolving, and the importance of *teaming* in tool development is growing. This paper has analysed developer-centred studies through the lens of HMT and one well-established team effectiveness model. The findings presented show that the use of these concepts is in its infancy in AI4SE and that there are several gaps for future research to address. They also point towards a number of implications and recommendations. We encourage AI4SE researchers and developers to consider the implications highlighted to apply HMT in this context.


## Acknowledgment

This work was supported by UKRI/EPSRC EP/T017465/1 and Lero, Ireland, grant 13/RC/2094_P2.


## References


[1] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *Journal of Systems and Software*, vol. 180, p. 111031, 2021.

[2] J. Rix, "From tools to teammates: Conceptualizing humans' perception of machines as teammates with a systematic literature review," *55th Hawaii International Conference on System Sciences*, 2022.

[3] J. B. Lyons, K. Sycara, M. Lewis, and A. Capiola, "Human–autonomy teaming: Definitions, debates, and directions," *Frontiers in Psychology*, vol. 12, p. 589585, 2021.

[4] J. Cleland-Huang, T. Chambers, S. Zudaire, M. T. Chowdhury, A. Agrawal, and M. Vierhauser, "Human–machine teaming with small unmanned aerial systems in a mape-k environment," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 19, no. 1, pp. 1–35, 2024.

[5] M. Mayer, "Trusting machine intelligence: artificial intelligence and human-autonomy teaming in military operations," *Defense & Security Analysis*, vol. 39, no. 4, pp. 521–538, 2023.

[6] V. E. Houston, B. A. Barrows, W. J. Manuel, and L. R. Le Vie, "Machine learning algorithms to improve model accuracy and latency and human-autonomy teaming," in *AIAA 2018 Aviation Forum*, 2018.

[7] D. Lo, "Trustworthy and synergistic artificial intelligence for software engineering: Vision and roadmaps," *preprint arXiv:2309.04142*, 2023.

[8] E. J. De Visser, R. Pak, and T. H. Shaw, "From 'automation'to 'autonomy': the importance of trust repair in human–machine interaction," *Ergonomics*, vol. 61, no. 10, pp. 1409–1427, 2018.

[9] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[10] E. Salas, D. E. Sims, and C. S. Burke, "Is there a "big five" in teamwork?" *Small group research*, vol. 36, no. 5, pp. 555–599, 2005.

[11] D. Partridge, "Artificial intelligence & software engineering: a survey of possibilities," in *The Software Life Cycle*. Elsevier, 1990, pp. 375–385.

[12] R. Feldt, F. G. de Oliveira Neto, and R. Torkar, "Ways of applying artificial intelligence in software engineering," in *Proceedings 6th International Workshop on Realizing AI Synergies in SE*, 2018, pp. 35–41.

[13] H. Sofian, N. A. M. Yunus, and R. Ahmad, "Systematic mapping: Artificial intelligence techniques in software engineering," *IEEE Access*, vol. 10, pp. 51 021–51 040, 2022.

[14] C. Watson, N. Cooper, D. N. Palacio, K. Moran, and D. Poshyvanyk, "A systematic literature review on the use of deep learning in software engineering research," *TOSEM*, vol. 31, no. 2, pp. 1–58, 2022.

[15] Y. Yang, X. Xia, D. Lo, and J. Grundy, "A survey on deep learning for software engineering," *ACM Computing Surveys*, vol. 54, no. 10s, pp. 1–73, 2022.

[16] A. H. Mohammadkhani, N. S. Bommi, M. Daboussi, O. Sabnis, C. Tantithamthavorn, and H. Hemmati, "A systematic literature review of explainable ai for software engineering," *arXiv:2302.06065*, 2023.

[17] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *TOSEM*, 2023.

[18] A. M. Madni and C. C. Madni, "Architectural framework for exploring adaptive human-machine teaming options in simulated dynamic environments," *Systems*, vol. 6, no. 4, p. 44, 2018.

[19] N. J. McNeese, M. Demir, N. J. Cooke, and M. She, "Team situation awareness and conflict: A study of human–machine teaming," *Journal of Cog Eng and Decision Making*, vol. 15, no. 2-3, pp. 83–96, 2021.

[20] J. C. Walliser, E. J. de Visser, E. Wiese, and T. H. Shaw, "Team structure and team building improve human–machine teaming with autonomous agents," *Journal of Cog Eng and DM*, vol. 13, no. 4, pp. 258–278, 2019.

[21] M. Johnson, J. M. Bradshaw, P. Feltovich, C. Jonker, B. Van Riemsdijk, and M. Sierhuis, "Autonomy and interdependence in human-agent-robot teams," *IEEE Intelligent Systems*, vol. 27, no. 2, pp. 43–51, 2012.

[22] S. Dubrow and K. L. Orvis, "Human-machine teaming: What skills do the humans need?" in *The Interservice/Industry Training, Simulation and Education Conference Published Proceedings*, 2020, pp. 1–11.

[23] D. Strode, T. Dingsøyr, and Y. Lindsjorn, "A teamwork effectiveness model for agile software development," *ESE*, vol. 27, no. 2, p. 56, 2022.

[24] T. Dingsøyr, T. E. Fægri, T. Dybå, B. Haugset, and Y. Lindsjørn, "Team performance in software development: research results versus agile principles," *IEEE software*, vol. 33, no. 4, pp. 106–110, 2016.

[25] T. Dingsøyr and T. Dybå, "Team effectiveness in software development: Human and cooperative aspects in team effectiveness models and priorities for future studies," in *5th CHASE*. IEEE, 2012, pp. 27–29.

[26] O. B. Newton, S. Saadat, J. Song, S. M. Fiore, and G. Sukthankar, "Everyboty counts: Examining human–machine teams in open source software development," *Topics in Cognitive Science*, 2022.

[27] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "Categorizing bugs with social networks: a case study on four open source software communities," in *35th ICSE*. IEEE, 2013, pp. 1032–1041.

[28] C. Bird, D. Ford, T. Zimmermann, N. Forsgren, E. Kalliamvakou, T. Lowdermilk, and I. Gazit, "Taking flight with copilot: Early insights and opportunities of ai-powered pair-programming tools," *Queue*, vol. 20, no. 6, pp. 35–57, 2022.

[29] A. Ziegler, E. Kalliamvakou, X. A. Li, A. Rice, D. Rifkin, S. Simister, G. Sittampalam, and E. Aftandilian, "Productivity assessment of neural code completion," in *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, 2022, pp. 21–29.

[30] M. Hozano, N. Antunes, B. Fonseca, and E. Costa, "Evaluating the accuracy of machine learning algorithms on detecting code smells for different developers," in *International Conference on Enterprise Information Systems*, vol. 2. SciTePress, 2017, pp. 474–482.

[31] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? assessing the security of github copilot's code contributions," in *Symposium on Security and Privacy*. IEEE, 2022, pp. 754–768.

[32] S. Imai, "Is github copilot a substitute for human pair-programming? an empirical study," in *44th ICSE: Companion*, 2022, pp. 319–321.

[33] A. M. Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, "Github copilot ai pair programmer: Asset or liability?" *Journal of Systems and Software*, vol. 203, p. 111734, 2023.

[34] O. Asare, M. Nagappan, and N. Asokan, "Is github's copilot as bad as humans at introducing vulnerabilities in code?" *Empirical Software Engineering*, vol. 28, no. 6, p. 129, 2023.

[35] E. Jiang, E. Toh, A. Molina, K. Olson, C. Kayacik, A. Donsbach, C. J. Cai, and M. Terry, "Discovering the syntax and strategies of natural language programming with generative language models," in *CHI*, 2022, pp. 1–19.

[36] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *CHI Extended Abstracts*, 2022, pp. 1–7.

[37] S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," *Proceedings of OOPSLA*, vol. 7, pp. 85–111, 2023.

[38] N. M. do Nascimento, P. S. Alencar, and D. D. Cowan, "Artificial intelligence vs. software engineers: An empirical study on performance and efficiency using chatgpt." in *CASCON*, 2023, pp. 24–33.

[39] J. H. Klemmer, S. A. Horstmann, N. Patnaik, C. Ludden, C. Burton Jr, C. Powers, F. Massacci, A. Rahman, D. Votipka, H. R. Lipford *et al.*, "Using ai assistants in software development: A qualitative study on security practices and concerns," in *ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 2726–2740.

[40] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *18th EASE Conference*, 2014, pp. 1–10.

[41] C. Wohlin and R. Prikladniki, "Systematic literature reviews in software engineering," *IST*, vol. 55, no. 6, pp. 919–920, 2013.

[42] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[43] P. J. Lucas, J. Baird, L. Arai, C. Law, and H. M. Roberts, "Worked examples of alternative methods for the synthesis of qualitative and quantitative research in systematic reviews," *BMC medical research methodology*, vol. 7, pp. 1–7, 2007.

[44] N. Nguyen and S. Nadi, "An empirical evaluation of github copilot's code suggestions," in *19th MSR Conference*, 2022, pp. 1–5.

[45] F. F. Xu, B. Vasilescu, and G. Neubig, "In-ide code generation from natural language: Promise and challenges," *TOSEM*, vol. 31, no. 2, pp. 1–47, 2022.

[46] W. Mendes, S. Souza, and C. De Souza, "You're on a bicycle with a little motor: Benefits and challenges of using ai code assistants," in *17th CHASE*, 2024, pp. 144–152.

[47] H. Zheng, "Developers' attitudes toward the future of artificial intelligence," *European Journal of Humanities and Social Sciences*, no. 2, pp. 114–118, 2019.

[48] C. R. de Souza, G. Rodríguez-Pérez, M. Basha, D. Yoon, and I. Beschastnikh, "The fine balance between helping with your job and taking it: Ai code assistants come to the fore," *IEEE Software*, 2024.

[49] J. Melegati, N. Nascimento, R. Chanin, A. Sales, and I. Wiese, "Exploring potential implications of intelligent tools for human aspects of software engineering," in *IEEE/ACM 17th CHASE*, 2024, pp. 121–132.

[50] A. A. Bangash, H. Sahar, S. Chowdhury, A. W. Wong, A. Hindle, and K. Ali, "What do developers know about machine learning: a study of ml discussions on stackoverflow," in *16th MSR Conference*. IEEE, 2019, pp. 260–264.

[51] I. Schieferdecker, "Responsible software engineering," *The future of software quality assurance*, pp. 137–146, 2020.

[52] S. Ntoa, "Usability and user experience evaluation in intelligent environments: A review and reappraisal," *IJHCI*, pp. 1–30, 2024.

[53] T. Lopez, H. Sharp, M. Wermelinger, M. Langer, M. Levine, C. Jay, Y. Yu, and B. Nuseibeh, "Accounting for socio-technical resilience in software engineering," in *16th CHASE*. IEEE, 2023, pp. 31–36.

[54] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, "Motivation in software engineering: A systematic literature review," *Information and software technology*, vol. 50, no. 9-10, pp. 860–878, 2008.