A Mixed Reality Approach for Innovative Pair Programming Education with a Conversational AI Virtual Avatar

Gilda Manfredi*
gilda.manfredi@unibas.it
University of Basilicata
Potenza, Italy

Ugo Erra
University of Basilicata
Potenza, Italy
ugo.erra@unibas.it

Gabriele Gilio
University of Basilicata
Potenza, Italy
gabriele.gilio@unibas.it

ABSTRACT

Pair Programming (PP) is an Agile software development methodology that involves two developers working together on a single computer. However, the physical presence of two developers has become a challenge in recent years due to the pandemic, necessitating remote collaboration methods such as Distributed Pair Programming (DPP). DPP has been found to have similar benefits to in-person PP, but the issue of team compatibility remains unresolved. These are more evident in the educational field of Agile methodologies. To address these challenges, we developed a novel approach by creating a Mixed Reality (MR) application that enables users to learn PP with the assistance of a conversational intelligent virtual avatar. The application uses the HoloLens MR device and a Conversational Agent (CA) extension integrated into Visual Studio Code to provide suggestions for improving the code written by the user. The virtual avatar animates these suggestions, making it appear to speak and interact with the user in real time. This system aims to overcome the limitations of common DPP methods, allowing a single developer to learn and apply the PP methodology even when a human partner is unavailable.

CCS CONCEPTS

• Software and its engineering → Software development techniques; **• Social and professional topics → Software engineering education**; **• Human-centered computing → Mixed / augmented reality**.

KEYWORDS

pair programming, extended reality, artificial intelligence, conversational agents

ACM Reference Format:
Gilda Manfredi, Ugo Erra, and Gabriele Gilio. 2023. A Mixed Reality Approach for Innovative Pair Programming Education with a Conversational AI Virtual Avatar. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE '23), June 14–16, 2023, Oulu, Finland*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3593434.3593952

*All authors contributed equally to this research.

EASE '23, June 14–16, 2023, Oulu, Finland

ACM ISBN 979-8-4007-0044-6/23/06…$15.00
https://doi.org/10.1145/3593434.3593952

1 INTRODUCTION

PP is an essential part of extreme programming [3], an Agile software development methodology emphasising collaboration, feedback, and rapid iteration. In PP, two developers work side-by-side on a single computer to write, test, and debug code. This methodology is highly valued in the industry for improving code quality, reducing errors, and increasing productivity. Moreover, PP is also recognised for its educational benefits, making it a widespread technique in introductory programming courses.

These benefits include retention, enjoyment, course success, confidence, and program quality [9]. However, one of the prerequisites of PP is the physical presence of two software developers. This requirement has posed a significant challenge in recent years, as the entire population has faced a serious epidemic that has forced people to avoid physical contact with others. This limitation restriction has prompted researchers to seek a solution that allows developers to use the PP methodology remotely. These statements are confirmed by Xu *et al.* [23], which assessed that the trend of online learning had necessitated the use of innovative approaches in programming education, with DPP emerging as an important methodology.

This has become especially relevant after the pandemic, which compelled educators to adopt new technologies and educational strategies to support effective learning. Xu *et al.* [23] also proved that the technique of DPP has the same benefits as in-presence PP in terms of productivity and code quality improvement. Despite the potential for DPP to address distance issues and facilitate remote collaboration, other challenges still need to be solved. One such issue concerns the compatibility of developers engaging in DPP. Urai *et al.* [21] found that in cases where there were differences in ability, students with lower proficiency levels tended to feel discouraged and lose motivation during DPP sessions.

Similarly, Tsompanoudi *et al.* [20] observed that groups of students with similar programming grades reported greater satisfaction with the collaborative experience. Furthermore, Dou and He [7] noted that students are more likely to pair with peers they perceive to have similar or higher skill levels, resulting in higher team compatibility. However, if both students in a pair perceive themselves to possess low proficiency levels, this could lead to frustration.

Predicting and mitigating incompatibilities between team members is a persistent issue in both collocated and distributed PP. Finding a remote collaborator with whom to establish a productive relationship can be challenging, and the quality of the collaborator can significantly impact the effectiveness of the PP experience. It can also be challenging to ensure that the collaborator has the necessary skills and knowledge to contribute to the programming task, hindering the adoption of PP as an effective teaching technique.

Essentially, traditional in-person PP has the limitation of requiring the physical presence of two developers, which can be difficult to coordinate. Similarly, DPP also requires the presence of a human partner, even when they are remotely connected. Both methodologies face limitations related to compatibility issues between partners, communication challenges arising from time zones and/or cultural differences, and technical constraints associated with remote collaboration. To address these challenges, we developed a novel approach that utilizes MR technology and a conversational intelligent virtual avatar to enable users to learn the PP methodology. Our approach aims to provide a valuable solution to these limitations allowing a single developer to effectively learn and apply the PP methodology without the need for a human partner. The main contributions of our approach include:

- Utilizing the HoloLens[1] MR device to create an MR environment that enables users to view their physical computer alongside a virtual avatar. The use of MR technology also enhances the learning experience by allowing users to practice PP in a more engaging and interactive way.
- Incorporating a conversational virtual avatar into our MR application to provide guidance, suggestions, and feedback to the user, mimicking the interaction with a human partner. This enables users to receive contextual and relevant suggestions based on their level of knowledge and experience in real-time, enhancing the effectiveness of the learning process.
- Providing a more engaging and interactive learning experience for users, as they can practice the PP methodology in a realistic and immersive environment. Our approach is a valuable solution in situations where finding a suitable human partner for PP may be challenging.

The remainder of this paper is structured as follows: in Section 2, we review the state-of-the-art of PP, DPP, and CA. Section 3 provides an overview of our system, including the hardware and software components. In Section 4, we describe the CA extension developed for VS Code, which monitors the user's code and provides suggestions for improvement. Section 5 explains the CA MR application developed using UE5, which utilises the text suggestions generated by the CA extension to animate the virtual avatar. Section 6 proposes a preliminary evaluation study for our approach in a real teaching scenario. Finally, in Section 7, we conclude the paper by summarising the contributions of our work and discussing its potential implications for future research in this field.

2 RELATED WORKS

PP is a well-known technique in software development, where two programmers work together on a single computer to develop software [6, 22]. Such practice has improved software quality, learning improvement, and developer productivity. This was demonstrated with a qualitative meta-analysis of existing studies conducted by Salge and Berente [16] and the controlled experiment performed by Williams *et al.* [22]. In recent years, PP has also been applied in educational settings, where it has been proven to enhance student learning and engagement. Salleh *et al.* [17] proposed a systematic review of empirical studies that analyze the PP effectiveness in higher education computer science/software engineering courses. Their analysis found that PP helps in improving students' satisfaction and grades.

In recent years, the concept of Distributed Software Development (DSD) [11] has gained prominence. In this context, DPP has emerged as a variation of PP, allowing developers to collaborate remotely on software development. Many studies have been conducted to explore the efficacy of DPP. For example, Baheti *et al.* [2] conducted an experiment at North Carolina State University to analyze various work arrangements for student teams engaged in developing object-oriented software. The study involved teams operating in both collocated and distributed environments, with some teams practicing PP while others worked individually. The study findings suggest that DPP is a viable approach to software development. DPP has also been studied in education, where it has been used to teach programming to students in different locations. Hanks [8] found that students using a tool for distributed pairing performed as well in their programming course as those who physically met to pair. However, DPP has limitations including the risk of the dismissal phenomenon, which can interrupt collaboration, reduce performance and quality [4]. Effective communication and collaboration management and an appropriate platform are required to maintain quality and performance [18]. Other limitations concern the developers' compatibility [7, 20, 21], a large amount of time spent on verbal explanation [19], the amount of program used to do a simple task [10]. Kuttal *et al.* [13] showed that replacing one human member of the pair with a CA can address these issues.

CAs, also known as chatbots or virtual assistants, are computer programs designed to simulate human conversation through text or voice interactions. These agents use natural language processing (NLP) to interpret and respond to user queries and requests, often providing automated customer support, information retrieval, or entertainment [14]. Robe *et al.* [15] developed PairBuddy, a CA for PP that communicates with users through voice and text. As in our solution, PairBuddy is embodied by a 3D avatar that, while it makes the interface more human, does not guarantee a great sense of immersion. This is because it is not so realistic and is also embedded in the interface of the IDE. Instead, our avatar appears like a real person and is displayed in an MR environment presented by the Hololens device, which provides a more immersive experience. Furthermore, PairBuddy generates only test cases, its code recommendations do not precisely match the task, and it cannot directly contribute to the code. In contrast, our solution offers a more comprehensive and tasks-specific PP experience.

Embodied CA has been studied as a potential interface in eXtended Reality (XR) systems which include Virtual Reality (VR), Augmented Reality (AR), and MR. XR technologies create digital experiences that can simulate real-world environments, augment them with digital information, or blend physical and digital elements to create entirely new hybrid environments [5]. In the literature, some applications embody CA in an XR space. Kopp *et al.* [12] developed a CA based on VR technology that can help users with tasks related to virtual construction. Anabuki *et al.* [1] designed an MR application characterized by an anthropomorphic CA living in the 3D space. In contrast to these approaches, ours leverages a conversational virtual avatar to facilitate PP education, whereas

[1]https://www.microsoft.com/en-us/hololens

previous applications had different objectives. Furthermore, our approach utilizes a CA extension integrated into VS Code to monitor code and offer real-time suggestions. This extension is then integrated with an MR application on the HoloLens device, allowing the virtual avatar to interact with the user naturally and intuitively.

3 SYSTEM OVERVIEW

The system architecture comprises three main components: the CA, the Visual Studio (VS) Code editor, and the HoloLens MR device. The CA is embodied by a humanoid virtual avatar interacting with the user and providing recommendations and guidance as the user codes. The CA and VS Code are tightly integrated, with the CA extension monitoring what the user writes and providing suggestions for improvements. In detail, during the avatar navigator phase, the user leverages Visual Studio (VS) Code to program, and the virtual avatar "observes" the code written by the user and provides suggestions for improving it. During the avatar driver phase, the user asks the virtual avatar to generate code based on a specific topic or problem. The suggestions are generated by the OpenAI API[2] and are then transmitted to the HoloLens application written in Unreal Engine 5.1[3] (UE5). The UE5 application utilises these text suggestions to animate the virtual avatar, making it appear to speak and interact with the user in real-time, providing a more immersive experience compared to traditional conversational agents.

One of the system's key advantages is the MR environment created by the HoloLens device. This environment allows the user to view their physical computer and the virtual avatar in the same space, making switching roles as driver and navigator easier. Additionally, the MR environment enables a more natural interaction between the user and the virtual avatar, as the avatar can be positioned next to the user and communicate through the built-in speakers of the HoloLens device. The HoloLens application is developed using UE5, which provides a rich and dynamic MR experience. Communications between the editor and the HoloLens application occur via Local Area Network (LAN) connection, allowing for real-time transmission of code suggestions and ensuring that the virtual avatar's recommendations remain up-to-date. Overall, the system architecture and MR environment offer a more immersive way for users to learn and apply the PP methodology, even without a human partner.

4 CONVERSATIONAL AGENT EXTENSION FOR VS CODE

The primary tool required for PP is an editor that allows the driver and the navigator to write and analyze code, respectively. We adopt VS Code as the editor the user will use to write code and interact with the conversational avatar. This choice was based on the fact that developers widely use VS Code. Its functions can be extended with third-party extensions written in TypeScript programming language, making it an ideal choice for integrating with the OpenAI API, which also supports TypeScript. The familiarity with this editor helps developers adapt to the system more easily, as they can continue to use their preferred coding environment while leveraging the benefits of the CA extension.

[2]https://openai.com/blog/openai-api

[3]https://www.unrealengine.com/en-US/unreal-engine-5

We developed the CA extension that has multiple functions. Firstly, the CA extension analyzes the user's input and provides suggestions accordingly. In particular, the extension can operate in two modes: driver and navigator. In driver mode, the user, acting as navigator, requests the extension to produce code based on a specific topic or problem, such as finding the best algorithm for a given task, fixing a specific error message, or writing a test case. In navigator mode, the extension provides suggestions to the user, which acts as the driver, based on the code they are currently writing. In this case, if the user has written code with syntax errors, the extension could suggest possible corrections based on the context of the code and the user's input. This flexibility enables the developer to switch roles during the PP session as needed. For example, a less experienced developer may choose to start as the navigator to receive more suggestions and guidance, while a more experienced developer may prefer to start as the driver and rely on the system for code completion or best practice suggestions. Additionally, the extension could provide suggestions for code completion as the user types or suggest best practices in coding, such as more efficient or readable ways to write code. For instance, if the user writes a loop that could be optimized, the extension could suggest a more efficient implementation. To do this, the CA extension analyzes the user's input in real-time using VS Code's built-in event system, which listens for changes to the document and triggers the analysis process accordingly. The extension does not overwhelm the user with too many suggestions but instead provides just the right amount of feedback to help the user improve their code. Additionally, the user can configure the CA extension to send suggestions based on their preference, such as after writing a line of code, a method, or a class. Furthermore, the user can manually trigger the analysis process using commands provided by the VS Code extension, such as a keyboard shortcut or a menu option. The adaptability of the system, along with its context-aware suggestions, ensures that it caters to the developer's level of expertise and provides appropriate suggestions accordingly. After analyzing the user's text, the extension provides it as input to the OpenAI GPT 3.5 models, which provide suggestions accordingly. OpenAI's GPT-3.5 models are advanced language models based on deep neural networks and are part of the GPT family of models. They are trained on a massive amount of internet text data, allowing them to generate coherent and contextually relevant text based on given prompts. With billions of parameters, GPT-3.5 models are currently the largest publicly available language models, capable of capturing a wide range of language patterns and nuances. They can provide relevant and coherent suggestions or completions based on input prompts by taking into account the surrounding text or prompt. GPT-3.5 models are versatile and can be fine-tuned for specific tasks, making them applicable in various applications such as text generation, summarization, and question answering. The generated text from GPT-3.5 models closely resembles human-written text in terms of style, tone, and grammar, making it sound natural and coherent for various language generation tasks.

The suggestions provided by GPT-3.5 models are generated in natural language, making them easily understandable and adaptable to developers of different expertise levels. The system does not rely solely on generating code snippets but also provides explanations, corrections, and best practices in a human-readable form. This allows developers of different levels to benefit from the suggestions, regardless of their familiarity with specific programming languages or concepts. If the suggestions are written in natural language and not as code, the extension sends them to the application running on the HoloLens device. The suggestions are sent as JSON files through the LAN connection. Specifically, the CA extension is an external client that sends HTTP POST requests to the HTTP server running on the HoloLens device. The body of the POST request contains the suggestions' JSON file. The CA extension also receives the status of the request from the server.

Another important feature of the CA extension is the possibility for the user to choose its PP role. This can be done through an extension command that notifies the CA extension and the HoloLens application of the user's role. Figure 1 shows an example of a suggestion with the user as a navigator.

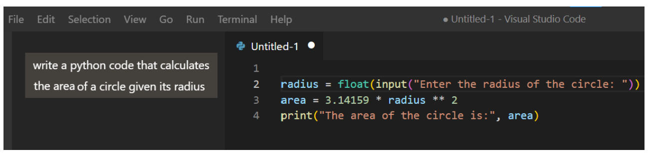


Figure 1: Example of suggestion with the user as navigator.

5 CONVERSATIONAL AGENT MR APPLICATION

The CA MR application is the server side of our system that runs on the HoloLens device. This application is developed in Unreal Engine 5.1, a game engine well known for its ability to generate realistic and immersive virtual environments. Necessary for our purposes is the Unreal Engine's MetaHuman[4] feature, which is a cloud-based creation system that allows for the creation of custom, high-quality character models within minutes. The MetaHuman can even be imported directly into UE5, making it an ideal choice for generating a realistic avatar for our MR application. In the MR environment of the CA application, the only virtual element present is the virtual avatar, which sits next to the user and engages in a conversation using the suggestions received from the CA extension for VS Code.

One of the challenges of creating our MR environment is correctly positioning the virtual avatar in the real world. To achieve this, the CA MR application uses the Scene Understanding modules of the HoloLens device. This module creates a spatial mapping of the real world, whose mesh is represented as planes classified based on their contextual interpretation, such as walls, floors, ceilings, and tables. With this module, we make the CA MR application able to track the position of the real floor and the desk where the user is sitting, allowing the virtual avatar to be correctly positioned relative to the user. The virtual avatar must also be animated to make the user's experience even more immersive. To create the virtual user's face animation, the suggestions received from the CA extension are first converted into text and then transformed into speech using a TextToSpeech plugin that integrates the Azure Speech Service[5] in UE5. The UE5 Sound Wave[6] of the speech is then used by the Metahuman SDK[7] to create a lipsync animation for the virtual user's face. The avatar's body assumes a sitting posture if it acts as the navigator and a sitting and typing posture if it assumes the driver role. In the last case, the avatar is also equipped with a virtual laptop. Figure 2(a) illustrates the user's point of view, showing their ability to view the computer screen with VS Code running on it, along with the virtual avatar. Figure 2(b) shows the entire MR environment, where the virtual avatar sits on the right side of the user, providing useful suggestions.

6 PRELIMINARY EVALUATION STUDY

The evaluation of our proposed application for learning PP was conducted as part of an ongoing research effort to address the persistent issues of compatibility and remote collaboration in both collocated and distributed PP contexts.

The evaluation followed a mixed-methods approach, combining quantitative and qualitative data collection methods. The evaluation was conducted in a real teaching scenario where university programming students were randomly assigned to experimental and control groups. The experimental group used our application for PP, while the control group followed traditional in-person PP methods. Data were collected through post-assessment surveys and coding assessments. Coding assessment scores were used to measure changes in participants' coding skills, while post-session surveys collected information on participants' satisfaction with the application's features, usability, and effectiveness in addressing the challenges of in-person and remote collaboration. Preliminary results indicate that participants who used our application for PP showed a statistically significant improvement in their coding skills compared to the control group ($p < 0.05$). Post-session surveys revealed high user satisfaction with the application's features, usability, and effectiveness in addressing the challenges of finding and collaborating with a human partner. These preliminary results suggest that our proposed application has the potential to address the challenges of Distributed and In-presence PP in a real teaching scenario. By avoiding potential incompatibilities between team members, our application can potentially enhance the adoption of PP as an effective teaching technique, especially in the context of remote work.

7 CONCLUSIONS

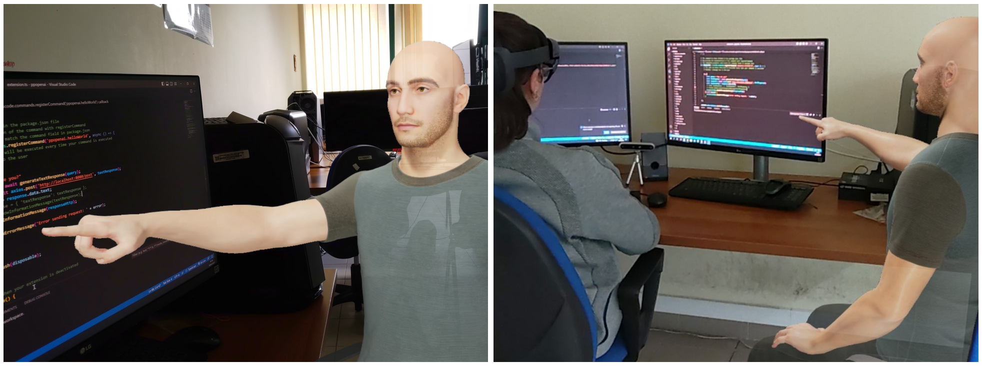
In this paper, we presented an MR educational solution for PP that leverages the assistance of a conversational intelligent virtual avatar to guide and provide suggestions to the user during the PP process. The proposed approach utilises the HoloLens MR device to create an immersive environment where the user can view their physical computer alongside a virtual avatar that serves as a navigator or driver, depending on the user's role.

[4]https://www.unrealengine.com/en-US/metahuman

[5]https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/overview

[6]https://docs.unrealengine.com/5.1/en-US/API/Plugins/SoundUtilities/USoundSimple/SoundWave/

[7]https://www.unrealengine.com/marketplace/en-US/item/66b869fa0d3748e78d422e59716597b6



(a) **(b)**

Figure 2: User's point of view (a) and MR environment with the conversational virtual avatar as navigator (b).

Our system's main innovation lies in integrating a CA extension into VS Code to provide real-time suggestions for code improvement, transmitted to the HoloLens app. The virtual avatar speaks and interacts with the user, creating an immersive learning experience. A single developer can learn and apply the PP methodology without a human partner in the MR environment, with the virtual avatar offering guidance as the user codes. The MR environment enables a natural interaction between the user and the avatar, enhancing the learning experience.

In conclusion, our MR educational solution for PP offers developers a novel way to learn and apply the PP methodology. In future work, we plan to conduct more extensive and controlled evaluations to obtain more robust evidence of our application's effectiveness. This includes investigating the long-term impact of the application on participants' learning outcomes to further refine the application based on the feedback received.

REFERENCES

[1] Mahoro Anabuki, Hiroyuki Kakuta, Hiroyuki Yamamoto, and Hideyuki Tamura. 2000. Welbo: An embodied conversational agent living in mixed reality space. In *CHI'00 extended abstracts on Human factors in computing systems*. Association for Computing Machinery, New York, NY, USA, 10–11.

[2] Prashant Baheti, Edward Gehringer, and David Stotts. 2002. Exploring the efficacy of distributed pair programming. In *Extreme Programming and Agile Methods—XP/Agile Universe 2002: Second XP Universe and First Agile Universe Conference Chicago, IL, USA, August 4–7, 2002 Proceedings 2*. Springer, Springer, Berlin, Heidelberg, 208–220.

[3] Kent Beck. 2000. *Extreme programming explained: embrace change*. addison-wesley professional.

[4] Gerardo Canfora, Aniello Cimitile, Giuseppe Antonio Di Lucca, and Corrado Aaron Visaggio. 2006. How distribution affects the success of pair programming. *International Journal of Software Engineering and Knowledge Engineering* 16, 02 (2006), 293–313.

[5] Nicola Capece, Gilda Manfredi, Vincenzo Macellaro, and Pietro Carratù. 2022. An easy Hand Gesture Recognition System for XR-based collaborative purposes. In *2022 IEEE International Conference on Metrology for Extended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*. 121–126.

[6] Enrico Di Bella, Ilenia Fronza, Nattakarn Phaphoom, Alberto Sillitti, Giancarlo Succi, and Jelena Vlasenko. 2012. Pair programming and software defects–a large, industrial case study. *IEEE Transactions on Software Engineering* 39, 7 (2012), 930–953.

[7] Wanfeng Dou and Wei He. 2010. Compatibility and Requirements Analysis of Distributed Pair Programming. In *2010 Second International Workshop on Education Technology and Computer Science*, Vol. 1. IEEE, Wuhan, China, 467–470.

[8] Brian Hanks. 2005. Student performance in CS1 with distributed pair programming. *ACM SIGCSE Bulletin* 37, 3 (2005), 316–320.

[9] Brian Hanks, Sue Fitzgerald, Renée McCauley, Laurie Murphy, and Carol Zander. 2011. Pair programming in education: A literature review. *Computer Science Education* 21, 2 (2011), 135–173.

[10] Brian F Hanks. 2004. Distributed pair programming: An empirical study. In *Extreme Programming and Agile Methods-XP/Agile Universe 2004: 4th Conference on Extreme Programming and Agile Methods, Calgary, Canada, August 15-18, 2004. Proceedings 4*. Springer, Springer, Berlin, Heidelberg, 81–91.

[11] James D Herbsleb and Deependra Moitra. 2001. Global software development. *IEEE software* 18, 2 (2001), 16–20.

[12] Stefan Kopp, Bernhard Jung, Nadine Lessmann, and Ipke Wachsmuth. 2003. Max-a multimodal assistant in virtual reality construction. *KI* 17, 4 (2003), 11.

[13] Sandeep Kaur Kuttal, Jarow Myers, Sam Gurka, David Magar, David Piorkowski, and Rachel Bellamy. 2020. Towards designing conversational agents for pair programming: Accounting for creativity strategies and conversational styles. In *2020 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE, IEEE, 1–11.

[14] James Lester, Karl Branting, and Bradford Mott. 2004. Conversational agents. *The practical handbook of internet computing* 1 (2004), 220–240.

[15] Peter Robe and Sandeep Kaur Kuttal. 2022. Designing pairbuddy—a conversational agent for pair programming. *ACM Transactions on Computer-Human Interaction (TOCHI)* 29, 4 (2022), 1–44.

[16] Carolina Alves De Lima Salge and Nicholas Berente. 2016. Pair programming vs. solo programming: What do we know after 15 years of research?. In *2016 49th hawaii international conference on system sciences (hicss)*. IEEE, IEEE, Hawaii, 5398–5406.

[17] Norsaremah Salleh, Emilia Mendes, and John Grundy. 2010. Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering* 37, 4 (2010), 509–525.

[18] Giuseppe Scanniello and Ugo Erra. 2014. Distributed modeling of use case diagrams with a method based on think-pair-square: Results from two controlled experiments. *Journal of Visual Languages & Computing* 25, 4 (2014), 494–517.

[19] David Stotts, Laurie Williams, Nachiappan Nagappan, Prashant Baheti, Dennis Jen, and Anne Jackson. 2003. Virtual teaming: Experiments and experiences with distributed pair programming. In *Extreme Programming and Agile Methods-XP/Agile Universe 2003: Third XP Agile Universe Conference, New Orleans, LA, USA, August 10-13, 2003. Proceedings 3*. Springer, Springer, Berlin, Heidelberg, 129–141.

[20] Despina Tsompanoudi, Maya Satratzemi, and Stelios Xinogalos. 2013. Exploring the Effects of Collaboration Scripts Embedded in a Distributed Pair Programming System. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (Canterbury, England, UK) *(ITiCSE '13)*. Association for Computing Machinery, New York, NY, USA, 225–230.

[21] Tomoyuki Urai, Takeshi Umezawa, and Noritaka Osawa. 2015. Enhancements to Support Functions of Distributed Pair Programming Based on Action Analysis. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania) *(ITiCSE '15)*. Association for Computing Machinery, New York, NY, USA, 177–182.

[22] Laurie Williams, Robert R Kessler, Ward Cunningham, and Ron Jeffries. 2000. Strengthening the case for pair programming. *IEEE software* 17, 4 (2000), 19–25.

[23] Fan Xu and Ana-Paula Correia. 2023. Adopting distributed pair programming as an effective team learning activity: a systematic review. *Journal of Computing in Higher Education* (2023), 1–30.