Pair-Coding as a Method to Support Intercoder Agreement in Qualitative Research

Jeffrey W. Paul
Price Faculty of Engineering
University of Manitoba
Winnipeg, Canada
umpaul59@myumanitoba.ca

Renato Bezerra Rodrigues
Price Faculty of Engineering
University of Manitoba
Winnipeg, Canada
bezerrar@myumanitoba.ca

Jillian Seniuk Cicek
Price Faculty of Engineering
University of Manitoba
Winnipeg, Canada
Jillian.SeniukCicik@umanitoba.ca

Abstract – The goal of this WiP paper is to provide an overview of our adaptation of pair programming from agile software development to support intercoder agreement in qualitative analysis. In pair programming, two programmers work together on the same program: one writes while the other observes and guides. Pair programming interleaves development and inspection activities and has been shown to produce higher quality software in a shorter time than individuals working alone. Our adaptation of agile software development pair programming to qualitative research, which we have called pair-coding, uses a similar approach to merge the qualitative research activities of coding and consensus-building by having two team members work on the same text simultaneously. In using pair-coding in qualitative research, the active team member highlights passages and assigns codes while the other team member observes and guides. Our research team anecdotally found that pair-coding of qualitative data provided benefits that were similar to the benefits of pair programming. Specifically, consensus and consistency were built continuously rather than at discrete stages. As well, differences in biases and worldviews were overtly revealed in the act of coding, thus improving the bracketing of biases. Finally, we found that consensus was better understood when built in the moment of coding rather than in comparison after the fact. We believe pair-coding could be effective in supporting the trustworthiness and credibility of qualitative analysis.

Keywords – qualitative methods, intercoder agreement, pair-coding, agile methods

I. INTRODUCTION

Qualitative methods in engineering research and engineering education research are growing in both use and acceptance [1]. A key step in many qualitative methodologies, such as grounded theory [2] and thematic analysis [3], is coding the data. Coding[1] in qualitative research and analysis is used to aggregate qualitative data (such as narrative text, images, audio, etc.) into smaller categories of information, called codes [4]. Each code is a "word or short phrase that symbolically assigns a summative, salient, essence-capturing, and/or evocative attribute for a portion of language-based or visual data" [5]. Codes are used to build meaning/make sense of the data. The researcher that codes the qualitative data is usually referred to as a coder.

To address the subjective nature of qualitative analysis, many methodologies propose using multiple coders who independently code the data and then meet to compare and build a common consensus. Creswell [4] refers to this as intercoder agreement. However, there is little direction on how to achieve intercoder agreement [4, p. 253]. Creswell lays out a process in which coders independently code three transcripts and then meet to discuss and consolidate their ideas, creating an initial codebook. They then separate again to independently code an additional two or three transcripts using the initial codebook and then meet again to discuss. This iterative process repeats as required throughout the project. In Creswell's example, the project achieved 80% agreement (as defined by assigning the same code to the same passages). This approach of independent expert analysis followed by comparison is not unlike the methods used to build a Natural Language Processing (NLP) gold standard corpus [6]. However, the goal in qualitative coding is generally to gain a broader understanding of the qualitative data, whereas the goal in developing an NLP corpus is to create a very specific understanding and annotations for automated processing.

The first two authors of this paper are Ph.D. students in Engineering Education, both with positivist backgrounds. Hence, to gain competency in qualitative research, we initiated a qualitative project, which has been published at the ASEE in the ERM division [7]. We initially followed a process similar to that proposed by Creswell [4]; however, we encountered some challenges in this process, which will be discussed later in the following sections.

Due to our pragmatic nature and the first author's experience in software engineering, and with the support of the third author (a qualitative researcher), we modified an agile method known as pair programming [8] to address some of the challenges we faced with our qualitative research. In *pair programming*, two programmers work together on the same program: one writes

[1] *Note*: the term coding has multiple meanings and in order to avoid confusion, in this paper we use the term *coding* to represent this aspect of qualitative research. Though the term can also refer to the act of writing software, we will not use the term in that manner, instead we will use the term *programming* to refer to the act of writing software.

while the other observes and guides. Pair programming interleaves development and inspection activities and has been shown to produce higher quality software in a shorter time than individuals working alone with discrete goals [8]. In our study, we adapted agile software development pair programming for qualitative research by having two team members work on the same data simultaneously, merging the qualitative research activities of coding and consensus-building. We call this approach *pair-coding*. Rather than coding the qualitative data independently and then meeting afterward to build intercoder agreement, we combined these tasks: the active team member highlighted passages and assigned codes to the data while the other team member observed and guided.

This method of having two coders work together on the same data, at the same time, appears unique in our literature review of qualitative research. For example, [9], [10] and [11] provide advice similar to Creswell and focus on team or collaborative approaches to consensus building, but each still requires independent coding before iterative meetings to build consensus or intercoder agreement. As well, neither of the expert qualitative researchers in the Faculties of Education and Nursing whom we consulted at our institution had an awareness of others using pair-coding or a similar technique in qualitative research.

Given how effective we anecdotally found our adapted approach in supporting and streamlining our qualitative analysis, we felt compelled to submit this work-in-progress paper prior to completing a formal research study on its strengths and weaknesses. Our goal is to present our anecdotal findings and critical reflections on pair-coding as a qualitative method for data analysis to initiate a conversation with the FIE community. As we have found no evidence of this approach being used in qualitative research, we feel it would be valuable to explore if the community has experience or knowledge of this or similar approaches before developing this method further.

This paper is presented in four sections. First, we will provide an overview of qualitative coding and the importance of intercoder agreement. Next, we will give a brief overview of the agile method known as pair programming to familiarize the reader with the genesis of our research. The third section will describe how we adapted pair programming to pair-coding. In the last section, we will provide an overview of our qualitative project [7] and our initial insights and critical reflections on pair-coding.

II. Qualitative Coding and Intercoder Agreement

Coding, and qualitative analysis, is more than writing "a good story, and finding a few wonderful quotes to throw in" [12]. Keep in mind that we are using the term coding here to mean qualitative coding – i.e., a way to aggregate qualitative data into smaller categories of information - not writing software. Effective coding *must* capture the voices of the participants and the salient themes of the narrative. To assure consistency and trustworthiness of the findings, qualitative researchers will often use triangulation of methods, researchers, and data sets. Thus, even on small projects, increasing the number of coders can help improve the credibility of the results. However, it is necessary to ensure that the way the data is interpreted is consistent among all the coders.

As Creswell [4] describes, intercoder agreement does not mean that the various coders assign the same code to the exact same snippet of text, but rather that the same code is assigned to the same general passage. That is, do coders see the same concepts in the data at the same places? This intercoder agreement attends to the subjective nature of qualitative research, which increases trustworthiness of the findings. Creswell goes on to stipulate an agreement of 80% is necessary for effective qualitative research, though he does suggest other statistical tools and measurements can be used.

For example, most Computer-Aided Qualitative Analysis Software (CAQDAS) tools such as NVIVO and MAXQDA have built-in statistical tools and can calculate inter-coder reliability, which is a mathematical measurement of coder agreement. These tools allow the researchers to focus on aspects of the text where there is low consensus. Though this automates the identification, the coders must still collaborate to address the differences. In the end, irrespective of the metric used, the goal is to provide a concrete metric addressing the subjective measure of qualitative research.

III. Agile Overview: Pair Programming

Agile software methodologies [13] originated in the 1990s due to the frustrations with failures attributed to heavy-weight, document-driven, waterfall methodologies [14]. The agile software philosophy has proven so successful that it has infiltrated other disciplines, such as Agile Project Management [15] and Agile Instructional Design [16].

One method within agile is known as *pair programming*. Although the concept of pair programming existed informally much earlier [8], it was popularized and formally defined in Extreme Programming in the late 1990s [17]. In pair programming [18], two programmers work jointly to produce software. One programmer, known as the *driver*, writes the software (i.e., works on the problem). In contrast, the other programmer, known as the *navigator* or *observer*, continuously observes, looking for defects, consistency, alternatives, etc. (i.e., works outside the problem). The two programmers regularly switch roles so both can have an insider and outsider perspective on the problem. In larger projects, programmers may rotate through multiple different pairings.

At first, it may seem counter-intuitive that having two people working on one section of software at the same time can be more efficient than having each work on their own section. However, pair programming has been shown to produce higher-quality software in shorter times than traditional independent programming [19]. Summaries of studies on pair programming can be found in [20] and [21]. The arguments for this method focus on continuous communication, continuous inspection, greater shared understanding of the project, and the benefit of integrating diverse strengths of the team members. This focus on individuals and their interactions versus the work product is a core principle of agile methodologies.

However, pair programming is not suitable for all problems or situations. The first author has been involved in ad-hoc successful agile software development projects that used informal pair programming. This experience underscored the value of the method, but it also identified some weaknesses that

are supported by research and practitioners. For example, successful pair programming is dependent on the willingness of the partners; having someone watch over your shoulder while you program is not comfortable for everyone [22]. Also, [23] found that pair programming was not as effective as individual programming on maintenance or change programming.

There are multiple ways to construct teams in pair programming, such as expert-expert, expert-novice, and novice-novice [24]. Novice-novice pairing brings specific benefits and challenges. For example, even though novice-novice pairings can be more effective than two novices working independently, it is generally discouraged as it is harder for novices to develop effective habits without an expert role model [24]. Remote pair programming (or virtual pair programming), where the two partners interact virtually using collaborative tools and screen sharing, is effective. However, research indicates that coordination can be more complex, and supplementary virtual collaboration tools are often required to support the project [25]. Given that agile methods encourage lightweight tools and simplicity, these additional tools can conflict with the underlying philosophy since it adds additional difficulty.

IV. Qualitative Method: Pair-Coding

In adapting pair programming to pair-coding for a qualitative research study, we focused on its fundamental philosophy: both team members working on the same transcript together, where one was the driver and the other the navigator. The driver's responsibility is to read, highlight and code the transcript, and pause the process to write memos as required – as done in a "normal" coding process. Meanwhile, the navigator focuses on querying the driver as to why they make certain decisions and presenting options. The navigator, working from outside the problem, should also consistently compare the text being coded with the previous codes to ensure their consistency.

Overall, the navigator's role is to make the driver explicitly aware of the decisions made while coding. To accomplish this in a qualitative research setting, the navigator can, for example, ask the driver to explain why they used a particular code for a section of data, what they mean by a specific code, or if they think another code would also be appropriate for that section. The interaction between the driver and the navigator enables in-the-moment discussions and reflections about the data and the evolving codes and themes. This allows consensus to be built concurrently *with* the analysis of the text instead of retroactively as with regular consensus-building meetings *after* the analysis of the text.

Our approach to pair-coding resembled pair programming in other ways as well. First, the roles of driver and navigator were switched on a regular basis. Depending on the length of the text being coded, this can happen during the coding of a single document or for each new document. Second, as with pair programming, pair-coding is mentally taxing, so breaks need to be scheduled, and sessions should be limited to a few hours.

There were two key issues we had to address in this project. First, the act of coding in qualitative research is not as evident as the act of programming; the driver is not writing out their solution or thoughts. Thus, for the navigator to gain insight into the driver's thought processes, we instituted a "think-aloud" protocol similar to that used in human-computer interface testing [26], and had the driver read aloud the text and verbalize their thoughts and analysis. This allowed the navigator to follow the driver's thought processes and kept both coders focused on the same section of the transcript. In addition, the driver used the mouse to indicate what they were reading.

Second, due to pandemic restrictions, we were working remotely, effectively adapting remote pair programming. We used Microsoft Teams to collaborate. The driver shared their screen (which had NVIVO 12 running), which allowed the navigator to follow the process live. The benefit of Microsoft Teams is that the navigator can gain virtual control over the driver's mouse and keyboard. This allowed the navigator to interact with the software (and data) more easily and enabled switching of roles iteratively as planned. In addition, having the camera on provided the coders with a more personal interaction.

V. Trial Pair coding: Our Experience & Insights

Our research project was a qualitative thematic analysis of 27 audio transcripts. These audios were secondary data (i.e., they had been recorded as part of a virtual interview series and made publicly available) [27]. Following Braun and Clarke's [3] Thematic Analysis method, each of us, as the first step, independently coded the same 1/3 of the transcripts (selected at random). We then met to compare our coding using traditional consensus-building methods for these transcripts [28]. The coding was done using NVIVO 12, which allowed us to use the automated comparison feature to identify the similarities and differences in our coding. During the consensus-building meeting, we used our memos to help us recall our thoughts and feelings from when we were individually coding.

However, we encountered several challenges with the consensus-building process. For example, even though we bracketed our research biases and wrote memos while coding independently, it was hard to recall our exact thoughts and reasoning behind each choice or decision. Also, it was difficult to explain in detail or have a full grasp of what was meant by certain codes. This might have happened because we used an inductive approach to coding, which means that we developed the codebook as we analyzed the data (note, it would be interesting to investigate whether a deductive approach, which uses a pre-determined codebook, would present the same challenges). Another issue was the time needed for this process; in addition to the time to code independently, we had to meet and have lengthy discussions to come to a consensus about the coding process and the codebook.

One issue that became apparent in this early stage was the fundamental differences in our coding approaches. As Saldana argues, "coding is an interpretive art" [5], and he outlines 29 different coding methods. In our project, the first researcher tended to use descriptive codes to "document and categorize" [5] the various statements made in the text. In contrast, the second researcher tended to use in vivo coding, rooting the code "in the participant's own language" [5]. This made it difficult to compare our codes.

As well, even though we were using NVIVO 12, which has tools to compare the codes of two researchers, we still found the process onerous, time-consuming and mentally taxing. Every

code had to be confirmed. In this case, even when both researchers coded the same text, it was often named differently. Thus, often the focus was on selecting the most appropriate phrase to capture the code, and this discussion took time.

To address our challenges with consensus building, we randomly selected another 1/3 of the transcripts and used pair-coding – the adapted concept of pair programming from agile software methods, as outlined above. Overall, we found pair-coding more intuitive than coding individually. The nature of the discussions during the pair-coding were more in-depth and reflective than the equivalent discussions that occurred after individual coding. The process helped us stay focused on the research question since, when coding independently, we tended to code information that was irrelevant to our research question. This was much rarer during pair-coding as the navigator, who was working outside the immediate issue, had the big picture perspective, which helped keep the focus on the research question and maintain consistency in the codes. This anecdotal finding aligns with the continuous inspection aspect of pair programming.

Pair-coding also helped address other issues, such as selecting a name for a code and identifying our biases. For example, we were able to immediately and explicitly identify differences in our worldviews, which improved the bracketing of biases and our understanding of both the transcripts and the evolving themes. Essentially, pair-coding allowed us to learn from each other's strengths and construct meaning together – instead of building it by ourselves and then rebuilding it again later. This anecdotal finding aligns with the collaborative learning often associated with pair programming.

After completing the pair-coding, we had intended to analyze the final 1/3 of the transcripts using traditional independent coding and consensus-building techniques. That is because the idea of pair-coding was to create a better understanding of the data and align the way both coders worked, so we could go back to the standard method. After coding a few of these final transcripts, we met to discuss our evolving insights. Two more anecdotal findings emerged. First, we both "heard" each other's voices as we coded independently: our partner's opinions and querying helped guide us even when we worked independently. Second, the intense nature of the pair-coding sessions gave us a stronger and more consistent sense of the emerging themes. We felt this was more in-depth than the shared understanding gained during the initial consensus building. The consensus was better understood when built in the moment of coding rather than in comparison after the fact. In pair-coding, we were both witnesses to each other's meaning-making process. It is easier to explain our thoughts and reasoning while they were happening instead of remembering after the fact. Because we found pair-coding was more effective and efficient for us, we elected to complete the remaining transcripts using pair-coding rather than reverting to the traditional method as we had initially planned.

We admit that a confounding factor at this point could also be linked to our growing familiarity with the text and the emerging themes. As well, one could argue that as we "heard" each other's voices, we should be able to revert to traditional coding. However, we found that pair-coding was significantly less cognitively demanding than coding independently and consensus-building after. As well, we found that pair-coding was more time-efficient. The time for pair-coding alone was shorter than the time spent individually coding plus the time for consensus-building. These findings align with the shared-understanding benefits of pair programming. Note that in agile pair programming, this would align with the efficiency gain when comparing with programming plus inspection (i.e., higher quality software).

VI. Limitations

Pair-coding, like pair programming, does have some limitations. The first challenge was scheduling a time to work collaboratively. Though the total time is less, it is all spent together. Second, we found that for pair-coding to work, both coders must be comfortable with each other and open to being challenged and questioned. Not everyone we discussed this concept with was enthusiastic about having someone watch over them. Lastly, pair-coding, like pair programming, can be mentally exhausting since it requires the coders to both analyze the data and build consensus while maintaining a positive social interaction. The stress of this balancing of analytical and social skills has also been noted in the literature on pair programming [22]. Overall, despite these issues, we found pair-coding to be a successful and rewarding approach and particularly helpful as novice qualitative researchers.

VII. Conclusion & Next Steps

Our research team found that the adaptation of pair programming from agile methods to qualitative coding in a qualitative research study was successful. Pair-coding enabled a stronger and more consistent understanding of the data and the findings. In addition, it made our biases and worldviews more overt and easier to address. Overall, it was also more time-efficient than coding independently and, even though this method had its challenges (scheduling, trust, comfort, mental exhaustion), we deemed our data analysis more credible and rigorous due to this process.

For the next steps, we plan to formally investigate pair-coding as a qualitative research method. We will explore novice-novice, expert-novice and expert-expert pair-coding, as well as deductive qualitative analysis. We believe that even if a deductive approach is taken, an initial session of pair-coding would provide relevant examples of the codes in the specific data set that would ensure stronger intercoder agreement and help clarify potential edge cases or protocols. We intend to use inter-rater reliability as a metric to assess the efficacy of pair-coding in qualitative research versus standard consensus building. Finally, we aim to explore how researchers with different worldviews would benefit from this process. This could be especially relevant in engineering education, an interdisciplinary field where people from different backgrounds and epistemological and ontological paradigms work together.

Acknowledgment

We are thankful to the reviewers of this paper whose comments helped us clarify our work.

References

[1] J. M. Case and G. Light, "Framing Qualitative Methods in Engineering Education Research," in *Cambridge Handbook of Engineering Education Research*, A. Johri and B. M. Olds, Eds., New York: Cambridge University Press, 2013, pp. 535–550.

[2] B. G. Glaser and A. Strauss, *Discovery of Grounded Theory. Strategies for Qualitative Research*: Sociology Press, 1967.

[3] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006, doi: 10.1191/1478088706qp063oa.

[4] J. W. Creswell, *Qualitative inquiry and research design: Choosing among five approaches,* 3rd ed. Los Angeles: SAGE Publications, 2013.

[5] J. Saldana, *The Coding Manual for Qualitative Researchers*: SAGE Publications, 2015.

[6] "The Gold Standard in Corpus Annotation," in *5th IEEE Germany Student Conference*, Passau, Germany, 2014.

[7] R. A. B. Rodrigues, J. W. Paul, and J. Seniuk Cicek, "Entering the Discipline of Engineering Education Research: A Thematic Analysis," in *ASEE Annual Conference & Exposition*, Longbeach, CA, 2021.

[8] L. Williams, "Pair Programming," in *Making Software What Really Works, and Why We Believe It*, A. Oram and G. Wilson, Eds., Farnham: O'Reilly, 2011, pp. 311–328.

[9] K. M. MacQueen, E. McLellan, K. Kay, and B. Milstein, "Codebook Development for Team-Based Qualitative Analysis," *CAM Journal*, vol. 10, no. 2, pp. 31–36, 1998, doi: 10.1177/1525822X980100020301.

[10] K. A. R. Richards and M. A. Hemphill, "A Practical Guide to Collaborative Qualitative Data Analysis," *Journal of Teaching in Physical Education*, vol. 37, no. 2, pp. 225–231, 2018, doi: 10.1123/jtpe.2017-0084.

[11] M. A. Cascio, E. Lee, N. Vaudrin, and D. A. Freedman, "A Team-based Approach to Open Coding: Considerations for Creating Intercoder Consensus," *Field Methods*, vol. 31, no. 2, pp. 116–130, 2019, doi: 10.1177/1525822X19838237.

[12] J. M. Case and G. Light, "Panel 1: Qualitative Methods: CHEER UP - Cambridge Handbook of Engineering Education Research (Updated Perspectives)," Jul. 2 2020. [Online]. Available: https://docs.google.com/document/d/e/2PACX-1vTn7uYmQfF5bsixitYdg_C2I0mMJnq1JfDhCgRDae0stNVHW549OmnW3KrYu0VgGcsvU-ZgumOi5wud/pub

[13] K. Beck *et al., Manifesto for Agile Software Development.* [Online]. Available: https://agilemanifesto.org/

[14] N. Abbas, A. M. Gravell, and G. B. Wills, "Historical Roots of Agile Methods: Where Did "Agile Thinking" Come From?," in *Lecture Notes in Business Information Processing, Agile Processes in Software Engineering and Extreme Programming*, W. van der Aalst et al., Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 94–103.

[15] G. Chin, *Agile project management : how to succeed in the face of changing project requirements*: AMACOM., 2004.

[16] M. H. Willeke, "Agile in Academics: Applying Agile to Instructional Design," in *2011 AGILE Conference*, Salt Lake City, UT, USA, 082011, pp. 246–251.

[17] R. Jeffries, A. Anderson, and C. Henderson, *Extreme Programming Installed*. The XP Series: Addison-Wesley, 2001.

[18] J. T. Nosek, "The case for collaborative programming," *Commun. ACM*, vol. 41, no. 3, pp. 105–108, 1998, doi: 10.1145/272287.272333.

[19] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," *IEEE Software*, vol. 17, no. 4, pp. 19–25, 2000, doi: 10.1109/52.854064.

[20] K. M. Lui, K. A. Barnes, and K. C. Chan, "Pair Programming: Issues and Challenges," in *Agile Software Development*, T. Dingsøyr, T. Dybå, and N. B. Moe, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 143–163.

[21] N. Salleh, "A systematic review of pair programming research -initial results," in *Proceedings of NZCSRSC 2008*, New Zealand, pp. 151–158.

[22] S. Hiehn, *5 years of Pair Programming.* [Online]. Available: https://medium.com/@stevehiehn/5-years-of-pair-programming-a5f644f6fd09

[23] E. Arisholm, H. Gallis, T. Dyba, and D. I. Sjoberg, "Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise," *IIEEE Trans. Software Eng.*, vol. 33, no. 2, pp. 65–86, 2007, doi: 10.1109/TSE.2007.17.

[24] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," in *Proceedings of the First International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2000)*, 2000.

[25] B. Böckeler and N. Siessegger, *On Pair Programming.* [Online]. Available: https://martinfowler.com/articles/on-pair-programming.html#PingPong

[26] C. Lewis, "Using the "Thinking-aloud" Method in Cognitive Interface Design," IBM Thomas J. Watson Research Center RC 9265 (#40713), 1982. [Online]. Available: https://dominoweb.draco.res.ibm.com/reports/RC9265.pdf

[27] A. Johri, *CHEER UP - Cambridge Handbook of Engineering Education Research (Updated Perspectives).* [Online]. Available: https://bit.do/cheerupvideos

[28] Y. I. Cho, "Intercoder Reliability," in *Encyclopedia of Survey Research Methods*, P. Lavrakas, Ed., 2455 Teller Road, Thousand Oaks California 91320 United States of America: Sage Publications, Inc, 2008, p. 345.