The effects of pair-programming in introductory programming courses with visual and text-based languages

Patrick Korber
CSLEARN – Educational Technologies
University of Vienna, Faculty of Computer Science
Vienna, Austria
patrick.korber@posteo.de

Renate Motschnig
CSLEARN – Educational Technologies
University of Vienna, Faculty of Computer Science
Vienna, Austria
renate.motschnig@univie.ac.at

***Abstract*— This full research to practice paper investigates the setting of pair-programming focusing on its effects on students' learning to program when using a visual versus text-based programming language. In particular, we are interested whether there are any differences in learning outcomes and social/motivational factors to keep in mind when learners' make their initial steps in programming in a visual versus text-based language while employing a pair-programming approach. First a systematic literature review (SLR) was conducted, encompassing searches in the ACM Digital Library, IEEE Xplore/Electronic Library, LNCS (Springer), and Google Scholar. The results of the SLR informed the second step, namely the design of a qualitative study consisting of the evaluation of students' achievements, a questionnaire, and focus interviews with students at secondary level (K10). Students worked in pair-programming (PP) and solo-programming (SP) mode while using a visual language (MIT App Inventor) and a text-based language (Python), respectively. In a nutshell, we found that regarding achievement, students tended to earn more points in the PP setting. This tendency was more distinct in the text-based condition compared to visual programming. In both conditions, students responded that in the PP setting they had more fun solving the assignments and the PP setting had helped to solve problems faster. Aside of the scientific findings, the paper describes the didactic concepts used in order to inspire educators to engage in similar practice and research. With this contribution we aim to enrich educators' and researchers' repertoire of making introductory programming lessons more engaging and successful.**

Keywords—Pair programming, visual programming languages, text-based programming languages, systematic literature review

I. INTRODUCTION

There is a broad consensus that the educational value of computer science can be understood as a fundamental skill of the 21st century [1]. Learning to program is an essential part of basic IT education, whereas the concept of *algorithmization* is considered a fundamental concept of information technology. Many authors agree that the use of visual programming environments such as Scratch guarantees results in positive learning experiences for novice programmers, see for example [2]. However, especially in advanced programming lessons, it is essential to also work with text-based languages to get an impression of professional programming environments.

For both text-based and visual programming languages, the concept of pair-programming has been discussed as a promising approach in research communities as well as in practice-oriented settings. In particular, the positive effects of pair-programming on motivation of learners have been investigated closely. According to self-determination theory, intrinsic motivation is sparked by increasing competence autonomy, and relatedness [3]. Since these motivational sources are immanent in pair-programming, the potential effects in introductory programming classes could be substantial. This motivated the first author to implement pair programming in his classes (level K9). The positive experience led him to devote his master thesis to systematically studying the literature, implementing the insights gained, and researching the effects of pair- vs. solo programming in his own classes. This paper shares the most essential findings with the goal to make course- and research-designs as well as experiences accessible to like-minded researchers and educators and thus to contribute to the advancement of learning to program while concurrently developing crucial 21st century competences.

The research-goal of this work is to find out whether the effects of pair-programming on various factors (e.g. motivation, problem solving or troubleshooting) are different when using text-based and visual languages. Thus, the central research question of this work is: How do the effects of pair programming differ in introductory programming courses of a text-based and a visual programming language? To answer this question, we investigate the specific challenges of text-based and visual languages for novice

programmers as well as the role of the learner's previous knowledge. Additionally, we focus on potential differences in troubleshooting when working with text-based and visual languages. In summary, the findings of this work are intended to better understand how pair-programming can effectively be used in introductory programming courses employing both text-based and visual languages. Thus, our target audience are educational researchers as well as educators interested in pair- and solo-programming settings using visual and text-based languages.

In chapter II, we describe the results of a systematic literature review (SLR) which was carried out to get a full picture of existing studies concerning pair programming in education. In chapter III, we sketch the learning sequences and present the research design for gathering and analysing empirical data on pair-programming with novice learners. Chapter IV summarizes the findings and results of the coding challenge, the questionnaire as well as the focus interviews. Chapter V follows up with an interpretation of the findings, potential implications as well as its limitations our work and directions for further research. Finally, chapter VI concludes the paper.

II. Systematic Literature Review

To better understand the current state of research concerning pair-programming in education, a systematic literature review (SLR) was conducted. In contrast to a conventional literature research, an SLR provides more explicit and reproducible results and can be an important tool to get a comprehensive picture of the research topic [4]. We followed the suggestions of Okoli and Schabram [5], who defined four main steps of the process: planning, selection, extraction, and execution. In our case, we focused on literature on pair-programming in education with a special interest in factors such as motivation, communication, and troubleshooting. We decided to focus on the following databases: ACM Digital Library, IEEE Xplore/Electronic Library, LNCS (Springer), and Google Scholar. We developed the following search-string which provided good results in a first attempt of finding relevant literature:

(Pair-Programming OR Pair Programming) AND (Classroom) AND (Efficiency OR Motivation)

This search-string resulted in 585 hits which had to be manually evaluated to narrow down the search. We therefore restricted the search to studies which were published after the year 2000 and excluded all works which were not focused on pair-programming in an educational context. This reduced the number of articles to 41 which allowed us to analyse them in detail and see that a broad spectrum of relevant topics was discussed: Beside the role of learning and teaching, motivation of learners and social factors such as gender or communication were also a major topic. Furthermore, many studies discussed questions concerning the efficiency and effectiveness of pair-programming in education. Following the suggestion of Salleh, Mendes and Grundy [6], we then conducted a systematic assessment of the quality of the reviewed articles. Thus, a quality-matrix of all 41 articles was established. We found that most of the articles (34 of 41) received a *very good* or *good* rating in our matrix which means that the quality of the reviewed articles was overall satisfying. In the next step, the results of the reviewed articles on the topics mentioned above were systematically analysed. We found that most studies agree that pair-programming leads to effective and efficient programming lessons: 8 out of 13 studies concluded that pair-programming had a positive impact on efficiency and effectiveness. Only one study found that the impact was negative, and four studies did not come to a clear conclusion.

It is noteworthy that some authors questioned whether pair-programming leads to efficient programming experiences: Especially with younger students, some studies found that there was a potential for distraction when working in pairs. Furthermore, most studies (4 out of 6) agreed that pair-programming led to higher motivation among students, whereas no study found a negative impact or no impact at all. Two studies presented unclear results concerning the impact on motivation. All studies agreed that many social factors such as gender or communication strategies had to be considered when conduction pair-programming lessons in introductory programming classes. The results of the SLR can be summarized as follows:

- Pair-programming is effective, but not always efficient, see [7].
- Pair-programming positively impacts motivation and self-esteem of learners, see [8], [6] and [9].
- Matching of the pairs according to programming skills and personal attitude is key for successfully implementing pair-programming activities, see [10] and [11].
- Social factors such as gender, effects of successes and failures, distribution of workload and the influence of partner changes must be considered, see [12] and [13].

Overall, the results of the SLR strengthen the assumption that pair-programming can be a powerful tool in introductory programming classes. Based on the results of the SLR, we designed a learning sequence for both a visual and a text-based language and developed focus interviews and questionnaires to get empirical data on potential differences of pair-programming and solo-programming.

III. Didactical- and research design

Since this work focuses on the individual experiences of students with pair-programming, the empirical data related to these experiences is primarily of a qualitative nature. Accordingly, we developed a learning sequence for four programming lessons for visual and text-based programming in solo-programming and pair-programming, respectively. 14 K-9 students (age 15-17, 10 male and 4 female) with limited programming experiences with a visual language (Scratch) participated in our research. Before working on the learning sequences, the participants took a coding quiz on general algorithmic understanding and basic programming fundamentals such as variables and loops, and

the pairs were formed according to the results of the quiz and personal preferences. After completing the lessons, the first author gathered empirical data using questionnaires as well as individual focus interviews.

A. Learning sequences

We decided to use MIT App Inventor for visual programming and Python for text-based programming in our learning sequences. For each programming language, the first author developed two sequences: One for solo-programming and one for pair-programming. All concepts are based on a constructivist view on learning and follow an explorative, self-regulated approach [14]. In explorative learning, learners try to achieve learning goals through certain actions, but the learning process is not linear, and it is likely that the learners are confronted with various problems during the process [15]. We considered the explorative learning approach as suitable for our research because it focuses on the process of learning rather than on the outcome.

Following this approach, our concepts were designed in a way that students were encouraged to work on programs without formal instruction from the teacher. To avoid discouragement due to the inappropriate complexity of the tasks, we implemented scaffolding elements for each tutorial offering different support structures such as tips or expected results [16]. In all tutorials, we tried to provide meaningful and playful programming experiences: With the MIT App Inventor, the students programmed a "whack-a-mole" game (pair-programming) and a classic "pong" game (solo-programming). In Python, the participants worked on a "casino-program" (pair-programming) and a "library" (solo-programming). Please note that interested educators can email the first author to obtains the learning materials used in this study.

B. Questionnaires and Focus Interviews

Guided interviews are a popular method for qualitative research since there is a profound body of literature concerning the generation and evaluation of qualitative data using this method [17]. We decided to conduct focus interviews, a sub-method of guided interviews, with selected participants because we were convinced that narrowing the focus to our research interest would result in meaningful qualitative data. In our specific context, the focus was obvious: Visual and text-based code examples were used to focus on the subjective perceptions of the pair-programming units as well as the solo-programming units. This procedure has been called problem-centred interview, where the problem is synonymous with the focus of the interview [18]. Following the suggestions of Helfferich [17], we tried to find a balance between openness of the interview on the one hand and a clear structure on the other hand, while aiming to keep up the narrative flow of the interviewee.

Prior to the focus interviews with selected participants, all participants filled in a questionnaire on their experiences with solo-programming and pair-programming immediately upon completion of the respective units. The students provided their first names in the answer sheets of the questionnaire because we wanted to improve the selection of interviewees for the focus interviews using this information. Based on the evaluation of the questionnaires, the first author who also had taught the class conducted interviews with six participants. The focus interviews were conducted after the regular programming units and recorded with a digital audio recording device. In the first few minutes the students practically dealt with visual and text-based code snippets. In the second step, the focus was on the experiences with pair-programming based on the pre-defined questions on motivation, troubleshooting and effectiveness. All interviewees gave valuable insights regarding our research interest and provided meaningful qualitative data which was then evaluated and interpreted in the next phase of our study using qualitative content analysis [19] and [20].

The data collection took place over several weeks during the elective course C*oding and Technology* at a secondary school (level K9) in Lower Austria. After completing all concepts, interviews were conducted with selected participants based on availability of the students and results of the evaluation of their programs. To gather comprehensive qualitative data, we first evaluated the questionnaires and then tried to motivate students with different views on pair-programming and solo-programming to participate in the focus interviews.

In the first unit of our field research, we matched pairs according to the results of the programming quiz as well as personal preferences. After that, the participants worked individually or in pairs on the learning sequence for 8 units á 50 minutes in total (2 units for each learning sequence). The students handed in the results of their work (MIT App Inventor or Python program files) immediately upon completion of the tasks. The first author who taught the classes then reviewed these program files and provided verbal feedback to the students. In the next step, he rated the quality of these program files using a simple point-based approach: A maximum of 10 points could be reached for each program. Points were subtracted if certain requirements were not met (e.g. a specific function was not implemented) or parts of the program were not working at all.

IV. Results and Findings

A. Results from the coding challenge and questionnaire

Figure 1 summarizes the results for all students, grouped into the respective pair-programming teams. It must be noted that these results cannot be used for a quantitative analysis since the sample of 14 students is obviously not sufficient. However, we believe that the results can serve as additional information pointing to potential differences of pair-programming with visual versus text-based languages.

		Quiz	Visual PP	Visual SP	Text. PP	Text. SP
Team 1	S(f)1	100%	100%	100%	100%	70%
	S(f)2	83%	100%	80%	100%	60%
Team 2	S(m)1	83%	90%	80%	90%	70%

	S(m)2	83%	90%	90%	90%	60%
Team 3	S(m)3	83%	90%	90%	100%	60%
	S(m)4	83%	90%	90%	100%	50%
Team 4	S(f)3	83%	90%	60%	100%	60%
	S(f)4	67%	90%	70%	100%	60%
Team 5	S(m)6	67%	100%	70%	90%	50%
	S(m)5	67%	100%	60%	90%	40%
Team 6	S(m)7	67%	60%	40%	70%	50%
	S(m)8	50%	60%	50%	70%	30%
Team 7	S(m)9	n.	70%	50%	60%	50%
	S(m)10	n.	70%	50%	60%	50%
	Average		***86%***	***70%***	***87%***	***54%***
	Min.		***60%***	***40%***	***60%***	***30%***

Figure 1: Evaluation of the quality of the programs

It is noteworthy that our participants performed better when working in pairs: The average value of points achieved was 86% (visual) and 87% (text-based) in the pair-programming units and only 70% (visual) and 54% (text-based) in the solo-programming units. On average, the students achieved significantly more points in the pair-programming units, whereas the difference was much higher in the text-based units (with 33% difference between pair-programming and single-programming).

The minimum of the achieved points shows a similar picture: in both text-based and visual programming, the worst result was 60% in the pair-programming lessons. Note that the minimum was lower in the text-based units (40% for visual programming, 30% for text-based programming). Although these results do not (yet) allow to draw solid conclusions regarding performance-related effects of pair-programming in introductory programming lessons, we can observe a trend: Students have clearly achieved more points in pair-programming and this effect was much stronger in text-based programming lessons than in visual programming.

The evaluation of questionnaires showed that the participants generally found text-based programming more difficult than visual programming. With text-based programming, one out of 14 students fully agreed and 4 partly agreed that it was difficult. With visual programming no single student fully agreed that it was difficult and 3 partly agreed that it was difficult (see Figure 2). This is not surprising and corresponds to the outcome of other studies on introductory programming [6].

Students agreed that it was easier to follow the tutorial when working in pairs. However, there is a significant difference between text-based and visual programming: With visual programming tutorials, 6 of 14 participants found it easier to complete the tasks in pairs, whereas with text-based programming tutorials, 12 of 14 participants agreed to this statement. The first author also asked the participants whether they had more fun working alone or in pairs. The result was clear: 11 of 14 participants (text-based programming) and 10 of 14 participants (visual programming) had more fun working in pairs.

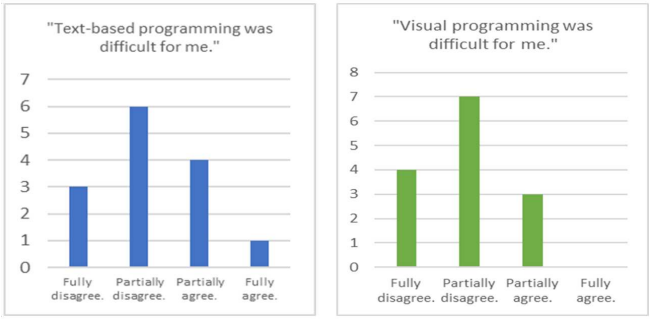

Figure 2: Difficulty text-based and visual programming

Much more surprising was the students' assessment of the learning effects of pair-programming and solo-programming: 8 of 14 students (text-based programming) and 10 of 14 stated that they learned more when working on their own. This self-assessment is relevant because we aimed to evaluate the effects of pair-programming on learning visual and text-based programming. For this reason, the first author conducted focus interviews to investigate this phenomenon in more depth. These are discussed after sharing the findings of the open questions concerning the effects of pair-programming.

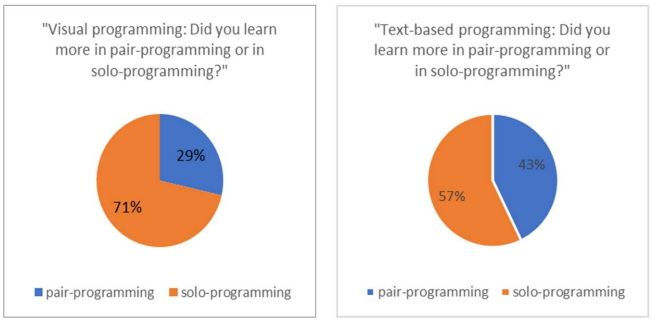

Figure 3: Effects on "learning"

In the open questions section, many students stated that one of the major advantages of pair-programming in both visual and text-based programming was the positive effect on motivation: "*I know that without my partner, I would have needed much longer. Besides, it was also funnier working in pairs.*" (S(f)1)[1] Some students pointed out that pair-programming was particularly helpful working on the text-based tutorials since it was easier to avoid syntax-errors:

"My partner drew my attention to typos, for example, and we both shared our knowledge and helped each other. It was rather funny and not depressing when we both didn't have a clue of where we are in the code and what the problem might be." (S(f)4)

The notion of S(f)4 that solving syntax-problems in Python was "*funny*" is remarkable. Visual programming languages are popular because syntax-errors are not very

[1] For reasons of anonymity, the name of each interviewee was coded. S(f)1 means student (female) number 1, S(m)1 student (male) number 1. We also translated the quotes from German to English.

likely to occur. It could be concluded that the positive effects of pair-programming on the motivation of programming novices is especially strong in text-based programming where troubleshooting syntax-errors is a common challenge.

However, it is also important to note that some students were quite sceptical concerning pair-programming. One student shared that unbalanced work-load could lead to dissatisfaction: *"Sometimes one person contributed more than the other because one of them didn't know what to do. This could lead to disagreements because everyone wants to finish it in their own way."* S(m)8. In the SLR, various studies were presented that deal with exactly this issue: Since pair-programming is a social act, there is always potential for conflict within the pairs. Making a special effort to match appropriate pairs has been described as key for solving this conflict, see [11]. It is particularly interesting that several students state that although pair-programming is more fun, the programming output is lower compared to solo-programming. *"You steer yourself from each other. Of course, you have more fun together, but that means that you slow down again."* (S(m)4)

The interview partners for our focus interviews were selected after the initial evaluation of the questionnaires as well as the programs created by the students. It was our aim to cover the broadest possible spectrum of experiences to get valuable data for our research interest. According to these criteria, the following teams were selected:

- Team 1 with S(f)1 and S(f)2
- Team 3 with S(m)3 and S(m)4
- Team 6 with S(m)7 and S(m)8

The interviews were conducted individually with each student and recorded for transcription and evaluation. We followed the suggestions by Kuckartz [21] for developing the categories and assign suitable passages from the transcribed interviews to our categories. Our main categories happened to mirror the research focus of this work. Figure 4 shows both the main categories and the subcategories used for coding the transcript.

	Category	Definition	Rules for Coding
MC1	Challenges with pair-programming (general)	General Challenges experienced when working on the projects.	All passages with explicit statements concerning challenges with MIT App Inventor
SC1.1	Challenges with MIT App Inventor	Specific challenges with visual programming	All passages with explicit statements concerning challenges with MIT App Inventor
SC1.2	Challenges with Python	Specific challenges with text-based programming	All passages with explicit statements concerning challenges with Python
MC2	General effects of pair-programming	General personal experiences of pair-programming effects	
SC2.1	Benefits pair-programming (general)	Favorable effects of pair-programming (general)	
SC2.2	Motivation through pair-programming (general)	Effects concerning motivation	
SC2.3	Disadvantages of pair-programming (general)	Negative effects related to pair-programming	
MC3	Differences of pair-programming with text-based and visual languages	Perceived differences of pair-programming with visual and text-based languages	Text passages where differences between text-based and visual programming are discussed
SC3.1	Differences (motivation)	Perceived differences concerning motivation	
SC3.2	Differences (quality of programs)	Perceived differences concerning the quality of the programs	
SC3.3	Differences (troubleshooting)	Perceived differences concerning troubleshooting	

Figure 4: Matching of categories

After matching suitable passages with the categories in Figure 4, we summarized and described the results.

B. Findings from focus interviews

MC1: Challenges with pair-programming (general)

"The problem with App Inventor is that there are so many functions […] but it isn't difficult, compared to Python." (S(m)3)

The results of the focus interviews provide a very clear picture of the challenges students faced. Only one student (S(m)8) found it rather difficult to work with the MIT App Inventor: "*With the App Inventor, it was sometimes extremely difficult to understand the relationships because you had to connect a lot, the images with the code and so on.*" (S(m)8) Given that this student did also face more challenges in the programming quiz and the other projects than his colleagues, this quote is particularly interesting. In contrast to S(m)8, most students noted that programming was rather easy with the MIT App Inventor. However, almost all students also described how difficult they found working with Python: "*I found Python to be more difficult because you have to focus a lot on typing errors.*" (S(f)2) These challenges were mentioned by most interviewees, which makes it clear that in the perception of the students, programming with Python is more demanding than programming with MIT App Inventor.

MC2: General effects of pair-programming

"I do believe that I was more motivated […]. We also motivated each other when something wasn't working." (S(m)3)

All kinds of effects of pair-programming were a common subject in our focus interviews and thus deserved an own category. Therein, statements addressing general effects of pair-programming were assigned to the correspondent subcategory (see Figure 4). Most statements corroborated both the results of the SLR and the evaluation of the questionnaires: Students mentioned that it was more fun working in pairs (S(m)3), that troubleshooting was easier (S(f)1), and that tasks were accomplished faster (S(m)7). These benefits were mentioned for both text-based and visual programming. The subcategory *SC2.2 Motivation through pair-programming (general)* was created because

many students described that working in pairs was more motivating than working alone. The following statement summarizes this experience: "*It's more fun because you can interact with the other person.*" (S(f)1) Notably, no single statement concerned potential negative effects of pair-programming on motivation.

The results of subcategory *SC2.3 Disadvantages of pair-programming (general)* are particularly interesting. Seven text passages were assigned to this subcategory and it is noteworthy, that all quotes in this category were contributed by students with a relatively high score in both the programming quiz and the evaluation of the programs. This means that students who experienced less difficulties with programming were especially critical concerning pair-programming. In summary, the following potential disadvantages of pair-programming were stated:

- pair-programming leads to "slow" progress (S(f)1)
- disagreements make it difficult to proceed (S(f)2)
- unequal learning pace leads to conflicts (S(m)3)
- more concentration when working alone (S(m)4)

The following statement seems particularly important in this context: *"Sometimes it annoyed me when I knew exactly how to do it […] and then M. wanted something different […] but in the end it worked with her version."* (S(f)2) The results of this subcategory show that the interviewees also see disadvantages of pair-programming, whereby especially those students who got along well with their project described these experiences.

MC3: Differences between pair-programming with text-based and visual languages

"Debugging is much more difficult with Python, together it is easier […] because there is simply a greater chance that you will find something." (S(m)3)

"With Python, it might be better to work together because you make more mistakes. But I think I did it pretty well on my own." (S(f)2)

Although the previous categories already indicate differences between text-based and visual programming when working in pairs, relevant text passages on this topic were collected in the subcategories *SC3.1 Differences (motivation)*, *SC3.2 Differences (quality of programs)* and *SC3.3 Differences (troubleshooting)*.

Five text passages were assigned to SC3.1, whereby all of them discussed the specific challenges of text-based programming and the advantage of working in pairs for solving these challenges: "*With Python, I wasn't that motivated alone […] I just got stuck. With App Inventor, I did better on my own."* (S(m)8) This quote is particularly relevant because, according to the evaluation of the programs, S(m)8 faced more problems than all his colleagues and achieved better results when working with his partner. The following quote of S(f)2 discusses a different aspect: "*I was motivated working with Python, it was really cool to do everything myself. I didn't really enjoy working alone with the App Inventor because it is even more fun to play around together.*" (S(f)2)

It is noteworthy that S(f)2 almost consistently achieved the full score in all projects. She describes the feeling of creating a program all by herself as particularly motivating. For her, solving the obstacles in text-based programming alone was a positive experience.

Concerning the quality of the programs (SC3.2), four text passages were assigned to this subcategory. The following quote by S(m)8 summarizes a notion that seems quite understandable: "*Overall, my Python programs were pretty bad, it just didn't work well. The program that we did together was better […] With the App Inventor, the difference wasn't that big.*" (S(m)8) In this quote, S(m)8 describes very specifically that for him, the perceived effect of pair-programming was much clearer in text-based programming than in visual programming. However, the following quote from S(f)1 seems like a complete contradiction to the statement of S(m)8: *"For me, it wasn't that much of a difference. It is almost the same, just a different layout. You just have to enter everything yourself and you don't have these blocks like with the App Inventor."* (S(f)1) This student did not personally experience any major differences. However, it is important to note that S(f)1 achieved the highest possible score in almost all her projects.

Six text passages were assigned to the third subcategory (SC3.3), whereas all quotes can be summarized by the following statement of S(m)8: "*My colleague helped me a lot with Python because he is […] not as chaotic as I am. It wasn't that bad with App Inventor, but with Python everything has to be very neat, otherwise nothing works.*" (S(m)8) This student had more problems with the didactic concept than his colleagues and describes the specific advantages of pair-programming for text-based programming quite clearly. With Python, the perceived advantages of pair-programming were significantly greater for S(m)8 compared to working with the MIT App Inventor. This statement can be found in different forms in almost all text passages in this subcategory, whereby it is noticeable that it was mainly students whose programs were rated with a medium to low number of points who commented on this topic.

V. Discussion

A. Interpretation and Implication of Findings

The research-goal of this work is to find out whether the effects of pair-programming on various factors like motivation, problem solving, or troubleshooting are different when using text-based versus visual languages. The results of our empirical study together with the findings of the SLR provided the framework to answer the central research question. The evaluation of the programs that students had created during our study showed that all pairs achieved better results in pair-programming than in solo-programming, whereby the effect was stronger with a text-based language. Together with the results of the SLR, we can distinguish at least three crucial differences between pair-programming with text-based and visual languages:

1) Effectiveness and Efficiency

Concerning the effectiveness (quality of the programs) it can be stated that pair-programming was more effective than solo-programming in both text-based and visual languages. Thereby the improvement of the quality of programs was more strongly evident in text-based programming. Nevertheless, the evaluation of the questionnaires showed that some students were critical towards the efficiency of pair-programming: "*Only one person can work and the other one sits next to this person and only gives instructions. You steer yourself from each other.*" (S(m)4) This student made an important point which was also stressed by other participants in both text-based and visual programming: 57% (visual programming) and 71% (text-based programming) of the students thought that they had learned more in solo-programming.

In the current study, the evaluation of the programs along with the results from our questionnaire and the focus interviews show that the students tended to have worked more effectively and efficiently in pair-programming. At the same time, significant differences between text-based and visual programming have been found: While high-performing students tended to perceive minor differences between text-based and visual programming in pair-programming, the effects were much more pronounced for students who struggled with the tasks. Those students who found the learning sequences particularly challenging benefited much more from pair-programming with text-based languages than with visual programming. This result seems very relevant for practitioners, especially those who work with students who find it difficult to learn to program.

It is important to note that the results of the SLR were equivocal. On the one hand, some studies questioned the efficiency of pair-programming, especially with younger learners. (see Faja 2014) On the other hand, we must look closely on the term *efficiency* since it is difficult to measure the efficiency of learning processes.

2) Motivation and Confidence

It is not surprising that in the empirical study we found that pair-programming positively impacted the motivation and confidence of programming novices. The results of the SLR showed similar effects for different target groups and contexts, see for example [6].

Regarding differences between text-based and visual programming, the evaluation of our focus interviews led to quite contrary experiences of the learners. On the one hand, we found that for students who struggled with the learning sequences, the effects of pair-programming on motivation and confidence were more pronounced in text-based programming. On the other hand, it seems that high-performing students can also experience that working with a more complex environment and typing code can be seen as a positive challenge that motivates them more than just visual programming.

This distinction can be significant for planning a successful introductory programming class: While for some students it seems that pair-programming is always the right choice in text-based programming lessons, others (especially high-performers) will probably appreciate elements of solo-programming in the didactical design. However, it must be stressed that in our study we found that also high-performing students enjoyed pair-programming in general.

3) Troubleshooting

It is evident that the importance of problem-solving increases as soon as students experience more obstacles in their tasks. The evaluation of the focus interviews showed that pair-programming was particularly useful to troubleshoot problems in text-based programming. The positive effects of pair-programming in this context are manifold: It is clear that *"[…] four eyes always [see] more than two*", as S(m)3 noted. In addition, troubleshooting in pair-programming has been described as funny, suggesting a higher level of motivation and more tolerance for frustration. Finally, it seems that for some students it is easier to systematically structure and organize their text-based code when working in pairs. Since visual programming languages provide a high level of structure, this positive effect turned out to be more pronounced in text-based languages.

Summarizing, the findings of this work provide evidence that pair-programming tends to lead to positive effects on confidence, motivation, and effectiveness in introductory programming classes for both text-based and visual languages. These findings corroborate the results from the SLR, in which 41 studies on this topic were systematically analysed [22].

The positive effect of pair-programming was particularly evident concerning the motivation and confidence of learners, which corresponds to the results of numerous studies: "*Almost all studies' findings reported that students' satisfaction was higher when using pair-programming compared with working individually."* [6, p. 11] It is important to note that this observation also confirms the SDT (self-determination theory) by Ryan and Deci, where *relatedness* is one of three basic needs or motivations [3].

However, the specific research question of this work was to explore potential differences of visual and text-based languages. In our study, the positive effects of pair-programming were found to be stronger in text-based languages. The rest of this section is devoted to discussing limitations of this work, its impact, and an outlook on further studies on pair-programming in education.

B. Limitations and Further Work

Although the questionnaires, programs, and focus interviews provided valuable insights into pair-programming with text-based and visual languages, various limitations must be mentioned. The evaluation of the programs created by the students does not allow a rigorous quantitative analysis due to the small sample size and potential bias inherent in the setup of the study. Further research, especially with quantitative methods and several teachers, could provide more solid outcomes for researchers and practitioners.

The findings of the empirical study also show that pair-programming leads to higher motivation and better results, especially with text-based programming. In this context,

further research on potential influences of complexity such as a systematic variation of complexity in the learning sequences as well as programming languages and environments would be interesting. Possibly, the perceived complexity of the tasks and programming environments corresponds with the subjective notion of success in pair-programming. It seems that particularly high-performing students considered the positive effects of pair-programming to be marginal when working on easy tasks.

Since all students in our study were participants of an elective class on Coding and Technology, it would be interesting to see whether the results still hold true when working with students in compulsory classes and other contexts. A further limitation lies in the fact that we were not able to investigate cultural, gender-specific, and special-needs related aspects in our study. The results of the SLR show that although various studies on gender and pair-programming were conducted, there is little evidence on the specific implications for practitioners [23]. It seems logical that the positive effects of pair-programming in introductory programming courses, especially with text-based languages, help to tear down gender-related barriers in programming, as some authors suggest [8]. Empirical studies investigating potential differences regarding gender-related issues could provide meaningful insights for both academic research and practitioners.

Finally, a further limitation of this work lies in the fact that although we tried to provide *optimal challenges* for all students in our learning sequences, there is room for improvement in the didactical design. Since our research suggests that the perceived complexity of the tasks is crucial for the personal experiences with pair-programming, the issue of *optimal challenge* seems to be fundamental. Further studies could focus on the impact of different didactical settings and learning arrangements on the effects of pair-programming. Moreover, studies following a research design like the one presented in this paper should explore the effects of pair-programming on students with different cultural backgrounds. Based on previous research on multicultural teams in education [24], we hypothesize that – under favorable conditions - pair-programming would result in positive effects of such teams regarding programming- as well as professional and intercultural competencies.

VI. Conclusions

The results from the SLR demonstrated that various success factors must be considered when implementing pair-programming in introductory programming classes. Building upon these insights, the results of this work strengthen the thesis that pair-programming leads to manifold positive effects in introductory programming classes. Investigating the differences between text-based and visual languages, we conclude that students who find it particularly challenging to learn a text-based language especially benefit from the positive effects of pair-programming.

These results have important implications for educators in the field of computing. Although students experienced positive effects like increased motivation and confidence and more effective programming with both text-based and visual languages, some differences became evident. Our empirical study provides further evidence that pair-programming can indeed be a powerful tool when some key factors are considered in the planning stage of the learning sequence. However, it is clear that these results cannot be generalized for all implementations of pair programming in the classroom.

The evaluation of our empirical data suggests that social factors are particularly relevant: Matching appropriate pairs is crucial for successful pair-programming activities: The personal relationship of the partners is essential, but at the same time it seems important to assign pairs according to programming experiences and general performance in computing. The evaluation of our data shows that especially for high-performing students it is important to create *optimal challenges* in pair-programming activities. Thus, positive effects of pair programming like increase of motivation and confidence can be fully developed for this group of learners.

For students who find introductory programming very challenging, the results of our research lead to a different conclusion: Here, the use of pair-programming is particularly promising, whereby an appreciative and clear communication – especially regarding possible mistakes and misunderstandings – seems to be a key factor. Combining these social skills with learning sequences that provide all students enough help and support for meaningful learning experiences, pair-programming is very suitable for this target group for introductory programming lessons.

Summarizing the conclusions of our empirical study, the potential of pair-programming in introductory programming classes seems evident. The evaluation of our data showed positive effects in both visual and textual languages. Nevertheless, some effects of pair-programming with text-based versus visual languages are different. This is why it is important to pay close attention to the target groups and their programming competencies when implementing pair-programming activities in introductory programming classes.

Acknowledgment

We are grateful for the support and understanding of teachers and administrative staff of BG/BRG Biondekgasse Baden. The empirical study was only possible because all students from the elective course Coding and Technology fully supported our work during and after class time. We are grateful to all of them for their time and openness. Thanks are also due to the school administration and the parents of our participants for their formal consent to conduct this study.

References

[1] S. Schubert and A. Schwill, Didaktik der Informatik. 2nd ed. Heidelberg: Spektrum Akademischer Verlag, 2011.

[2] M. Kalogiannakis, N. Zaranis, S. Papadakis et al, “Using Scratch and App Inventor for teaching introductory programming in secondary

education. A case study," Int. J. Technology Enhanced Learning, Vol. 8, no. 3, 2016, pp. 217-237.

[3] R.M. Ryan and E.L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," American Psychologist, Vol. 5, no. 1, 2000, pp. 68–78.

[4] A. Fink, Conducting Research Literature Reviews: From the Internet to Paper, 2nd ed. Thousand Oaks, California: Sage Publications, 2005.

[5] C. Okoli, and K. Schabram, "A Guide to Conducting a Systematic Literature Review of Information Systems Research," Sprouts: Working Papers on Information Systems, Vol. 10, no. 26, 2010.

[6] N. Salleh, E. Mendes, J. Grundy, "Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review," IEEE Trans Software Engineering, Vol. 37, no. 1, 509-525, 2011.

[7] S. Faja, "Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses," Information Systems Education Journal (ISEDJ), Vol. 12, no. 6, pp. 36-45, 2014.

[8] J. Liebenberg, E. Mentz, B. Breed, "Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT)," Computer Science Education, Vol. 22, no. 3, pp. 219-236, 2012.

[9] N. Nagappan, L. Williams, M. Ferzli, et al, "Improving the CS1 experience with pair programming," ACM Sigcse Bulletin, Vol. 35, pp. 359-362, 2003.

[10] K. Man Lui and K. Chan, "Pair programming productivity: Novice–novice vs. expert–expert," Int. J. Human-Computer Studies, Vol. 64, pp. 915-925, 2006.

[11] P. Maguire, R. Maguire, P. Hyland et al, "Enhancing collaborative learning using pair programming: Who benefits?," All Ireland Journal of Teaching and Learning in Higher Education, Vol. 6, no. 2, pp. 1411-1415, 2014.

[12] S. Campe, J. Denner, E. Green, et al, "Pair programming in middle school: variations in interactions and behaviors," Computer Science Education, Vol. 30, no. 1, pp. 1-25, 2019.

[13] J. Tsan, J. Vandenberg, Z. Zakaria et al (2020). A Comparison of Two Pair Programming Configurations for Upper Elementary Students, The 51st ACM Technical Symposium on Computer Science Education SIGCSE '20, (pp. 346 – 352).

[14] J.S. Bruner, On knowing: Essays for the left hand. Cambridge, Mass: Harvard University Press, 1967.

[15] M. Kerres, Multimediale und telemediale Lernumgebungen. Konzeption und Entwicklung, München: Oldenburg, 2001.

[16] D. Nam, Y. Kim, T. Lee (2010). The Effects of Scaffolding-Based Courseware for The Scratch Programming Learning on Student Problem Solving Skill, Proceedings of the 18th International Conference on Computers in Education ICCE2010, (pp. 723-727).

[17] C. Helfferich, "Leitfaden- und Experteninterviews," in *Handbuch Methoden der empirischen Sozialforschung*, N. Baur and J. Blasius, Eds. Wiesbaden: Springer VS, 2019, pp. 669-686.

[18] A. Przyborski, and M. Wohlrab-Sahr, Qualitative Sozialforschung, München: Oldenbourg, 2008.

[19] P. Mayring, Einführung in die qualitative Sozialforschung. Eine Anleitung zum qualitativen Denken, Weinheim und Basel: Beltz, 2002.

[20] P. Mayring, Qualitative Inhaltsanalyse, Grundlagen und Techniken, Weinheim und Basel: Beltz, 2010.

[21] U. Kuckartz, Qualitative Evaluation: Der Einstieg in die Praxis, Hamburg: VS-Verlag, 2008.

[22] P. Korber, "Die Effekte von Pair Programming im einführenden Programmierunterricht mit visuellen und textbasierten Programmiersprachen," master thesis, Universität Wien. Fakultät für Informatik, Vienna, 2020.

[23] B. Zhong, Q. Wang, J. Chen, "The impact of social factors on pair programming in a primary school," Computers in Human Behavior, Vol. 64, no. 1, pp. 423-431, 2016.

[24] R. Motschnig and E. Geiderer, "What Students Learn and Take Away from a Student-Centered Course on Communication and Professional Skills: Analyzing the Content of Students' Self-Evaluations," *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1-9, doi: 10.1109/FIE43999.2019.9028432.