


International Journal of Learning, Teaching and Educational Research
Vol. 23, No. 7, pp. 271-288, July 2024
<https://doi.org/10.26803/ijlter.23.7.14>
Received May 29, 2024; Revised Jul 15, 2024; Accepted Jul 18, 2024

Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University

Sithandiwe Twetwa-Dube 
Walter Sisulu University
Umtata, South Africa

Abstract. Many rural schools in the Eastern Cape (EC) face challenges in improving their standards. Almost 97% of first-year IT Diploma students at a selected University come from these rural EC schools. Teaching programming to these first-year and extended program students is difficult, leading to high dropout rates and students switching to other IT specializations. The course is considered difficult for them to grasp, causing anxiety and frustration. Computer programming learning requires high-level critical and logical thinking skills, which poses an even greater challenge. Many students lack basic computer skills and struggle with English, the medium of instruction, adding to their difficulties. These challenges, along with others, hinder the students' performance in the programming course. The researcher aims to explore how group work strategies could improve student performance. Students' confidence, effort, and communication abilities play a significant role in their success in these classes. This study seeks group work strategies to help first-year and extended program students understand computer programming. It collected data from 88 students (48 first-year and 40 extended program students) using questionnaires, employing mixed methods. The findings suggest that group work could help students from disadvantaged backgrounds better grasp computer programming. This mixed method was supported by the social constructivism theoretical framework.

Keywords: Computer Programming; Digital Skills; Social Constructivism; Group work strategies

1. Introduction

As technology continues to transform our lives, industries, and businesses rapidly, it is important to remember that the computer programmer at the core of any technological innovation is the person who creates the system (Susanti, 2021). Students must incorporate several elements when engaging in exploratory programming tasks or activities. For instance, Rosenberg-Kima et al. (2022) found that when teaching students to program or code, whole-task instruction is superior to part-task instruction because it teaches students how to integrate coding fundamentals into problem-solving techniques in addition to teaching them individually. According to the study by Ideris et al. (2019), groups of students should use Scratch software to tackle programming challenges. This instruction can help students become more proficient in higher-order thinking skills and exam scores. Furthermore, mental models, that is, frameworks that aid students in comprehending how their minds function and the reasons behind their thoughts, help learn CP and improve student's programming skills (Rosenberg-Kima et al., 2022).

Kovari and Katona (2023) demonstrate the impact that desire and self-efficacy have on student's success in programming. High-self-efficacy students are likelier to use positive learning strategies, select more challenging assignments, and set high standards in a learning setting (Rosenberg-Kima et al., 2022). The group strategy approach proved to be so interactive and exciting. Furthermore, some students could build relationships that aided their engagement, collaboration, and success in the chosen course; realizing that students would like observations to be presented principally and thoughtfully was also crucial (Khomokhoana, 2023). The fact that information technology is now built into many academic fields is hardly new. Professionals in various fields are realizing how important it is to have deep knowledge of Information and Communication Technology (ICT). Using the foundational ideas of computer science, computational thinking aligns with 21st-century skills in problem-solving, system design, and human behaviour analysis. Cognitive, linguistic, creative problem-solving, attitude, and collaboration abilities are among the skills that can be developed in computational thinking principles (Ng et al., 2023; Nouri et al., 2019).

Learning CP calls for both cognitive and metacognitive skills. To apply their creativity and solve difficulties, the student must understand the syntax and semantics of a chosen programming language. It combines logical thinking with creativity. Consequently, proficiency in CP is essential for advancing technology in all fields and is thus required for economic growth and national development. Computer programmer's skills will be necessary for developing nations to integrate ICT into their systems. The steps needed to solve problems when learning programming are problem identification and definition, planning, problem-solving design, coding, testing, and documentation. Due to changing views on the sense of constructivism and how it is put into practice in the classroom, first, the core of knowledge and how students develop meaningful knowledge is an essential topic. Good communication is important to understanding CP, whether you are South

African or not. Many students are confused by the various meanings of some English and computer terms. The best group work strategies for teaching CP are confirmed by some authors, yet language barriers remain a common problem.

According to the study by Costa et al. (2017), learning challenges related to introductory programming courses are a significant reason why many first-year students discontinue the rest of the course. Approximately 97% of first-year IT Diploma students in the selected University come from rural schools in the EC. Introducing programming as the content to these students (first-year and extended programme) is a challenge, resulting in a high dropout rate or a switch to another IT specialisation programme.

For most students unfamiliar with programming, learning the language is seen as a tough and demanding undertaking. Students who want to learn programming need good problem-solving abilities because failing a course can negatively impact their ability to understand the phases of algorithm creation (Ubaidullah et al., 2021). Many students entering the University lack basic computer and digital skills besides the English language as a medium of instruction used in programming. These challenges, coupled with others, hinder the knowledge of students in the CP course. Although some existing studies have attempted to address the challenge (Khomokhoana, 2023; Piwek & Savage, 2020). However, the results presented from these existing studies have not been able to achieve the expected outcome, especially among the rural-based institutions in Sub-Saharan Africa. Hence, this study explores how and why group work strategies can support first-year students' understanding of computer programming.

The contributions of this paper are outlined below:

- Around literature improvement
- Around policy maker guide
- Techniques to adopt in teaching rural-based first-year programming students.

The study aims to determine how group work strategies can enhance the conceptualization of computer programming for first-year students. It also demonstrates that these strategies support students understanding, making it easier for them to express their understanding in English, the language of learning and teaching.

The remainder of the paper is outlined in the following sections: Section 2 presents the literature review; the theoretical framework is presented in Section 3; the research methodology is described in Section 4; and Section 5 presents the data analysis and discussions. The paper is concluded in Section 6, and limitations, recommendations, and future research are described in Section 7.

2. Literature Review

Nisan and Schocken (2021) define programming, or computer programming, as the creation and execution of different sets of instructions to enable a computer to perform a specific function. The tasks are usually problems that require solving and creating a program is the outcome of these tasks. CP is a problem-solving technique that can be understood in the context of improving problem-solving abilities; it requires a broad understanding of programming languages, algorithms, and data structure (Biswas, 2023). It is a crucial course covered in undergraduate IT / ICT programs at Higher Education Institutions (HEI).

Programming students must become more innovative, creative, collaborative, and knowledgeable about data structures and algorithms (Nair, 2020). Peer interaction and group work discussions can make up the self-motivated and active learning environment indicated (Chetty & van der Westhuizen, 2017; Zhang et al., 2013). Communication becomes difficult because most university students come from diverse high schools with varied backgrounds, especially those from rural schools. English is typically taught as a second language in rural schools in South Africa despite being the primary language of instruction at universities (Lukose, 2021). Our everyday routines depend heavily on communication (Bygate, 1987), and through discussions, students can learn from one another, pick up new abilities, and enhance their confidence and motivation (Dörnyei & Thurrell, 1994). To provide an efficient solution, programming naturally requires a high level of critical thinking. This required using specific algorithms and programming principles (Cheah, 2020). The intention of teaching students CP is to learn the use of programming to solve problems, not to make them memorize programming language (Liu et al., 2018). Liu et al. (2018) explained how language barriers might make it difficult for students to follow sessions that are taught in a second or foreign language when trying to teach specific subjects. Bravo-Sotelo (2020) further explains that the teacher would employ a communication strategy called code-switching which combines two different languages to facilitate conceptual understanding.

CP has been a challenging topic to understand and master. The issue has a global scope and keeps worsening locally (Cheah, 2020; Robins, 2019). Even though there are many educational resources accessible to support the teaching and learning of CP, the issue still exists today. Introduction to CP courses had significant failure and high dropout rates (Chetty & van der Westhuizen, 2015). The student's deficiency in problem-solving abilities is one of the main difficulties in programming (Aissa et al., 2020). To address challenging real-world problems, students must understand computer programming beyond syntax and flow (Bosse & Gerosa, 2017; Piwek & Savage, 2020).

Most CP courses are now taught in-person in classrooms aided by online resources like Moodle, self-assessment quizzes at the end of each topic, group work practical assignments, and summative assessments, including final exams at the end of the syllabus. Through participating in group work activities, students can effectively

collaborate on class tasks and express their thoughts clearly. Based on the gathered literature, it can be argued that there is still limited research on how different ability levels of beginners from disadvantaged rural areas versus advanced students differ. These group work strategies are developed for mixed-ability groups to guarantee that every student benefits. It can be further observed that there are existing gaps in understanding how group work in programming courses affects students' collaboration and communication skills and some approaches to assess and improve soft skills within the context of programming education. There is limited research on how group work strategies impact diverse student populations, including underrepresented groups in technology. Not enough frameworks and alternative models can measure the success of group work strategies in achieving learning objectives. In addition to the aspects discussed in this section, further explanation is provided within the theoretical framework section of this study.

3. Theoretical Framework

The social constructivism theory developed serves as the theoretical underpinning for this study. Vygotsky (1978) defined the *more knowledgeable other* (MKO) as someone who has a higher level of understanding or ability in a specific area than the students themselves. Teaching CP through group work strategies requires a well-thought-out theoretical framework that considers the course's goals, the learners' needs, and the best practices in higher education. Group activities were supposed to be a setting where translanguaging may emerge and thrive. The current practices of students misinterpreting assignments, class exercises, and practical questions were problematized in the critical and logical reflection framework, and a plan of action was developed to change the practice. Figure 1 illustrates how active classes are being conducted.

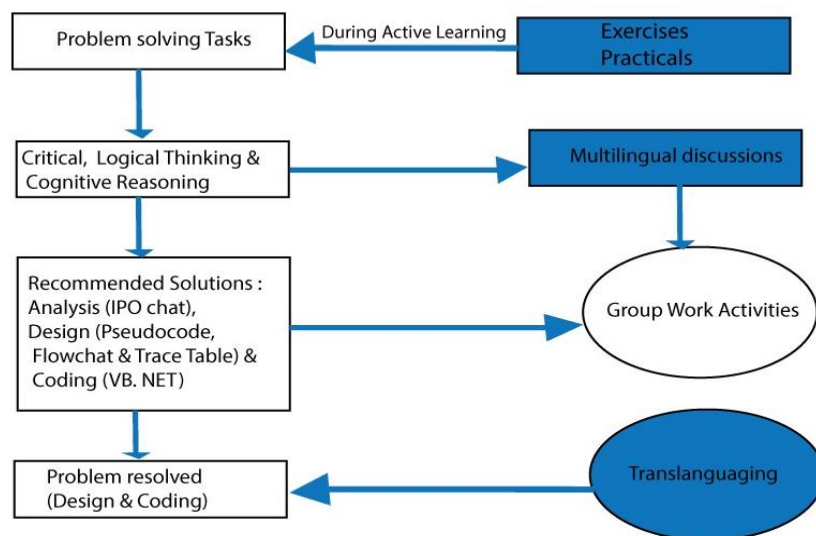


Figure 1: Framework for group work strategies to teach computer programming

Students were provided with course notes and tasked with weekly readings, problem-solving, and practical questions based on these notes. The lecturer initially facilitated group activities by scaffolding and modelling thinking strategies. Together, the class explored topics through a mix of group, individual, and lecturer inputs. After covering the theory, students collaboratively designed and created solutions to programming problems worked individually, and reviewed each other's solutions. Written answers were discussed in groups and as a class. The lecturer assigned students to groups based on assessment results to ensure a mix of skills. A multi-literacies approach was incorporated by initially encouraging the use of students' mother tongues, followed by English explanations.

Students were taught to analyze problems and write structured algorithmic solutions using pseudocode and visual basic VB.Net (Console Application). Groups of five to six were formed to facilitate collaboration and communication. The lecturer introduced new topics with presentation slides and guided students through programming tasks using a student module guide (SMG) and a lecturer's work schedule guide (LSG). Group members exchanged solutions and self-marked during class, enhancing understanding and enjoyment and forming part of the formative assessment.

Active learning involves collaboration and expertise-sharing to achieve common goals, enhance problem-solving skills, and foster innovative solutions. Effective communication within groups develops students' communication skills and ability to give and receive feedback, which is crucial for computer programming. This setup allowed students to start tasks confidently and manage difficulties independently. This study aimed to find the group work strategies that can support first-year students' understanding of CP and the use of the social constructivism theoretical framework aimed at assisting students to grasp computer programming. Development Software 1 is the compulsory course the selected University offers for all first-year students in all IT specializations. This course aims to equip students with a solid understanding of the basic principles of CP that relate to all CP languages. Also, the CP syllabus encompasses the theory and practical sessions using the VB.NET console application. The study's results are intended to contribute to and improve the communication, social skills, critical thinking, and cognitive skills of students pursuing computer programming at a first-year level. Furthermore, it provides opportunities for lecturers to adopt the recommended group work strategy to improve students' performance in computer programming.

4. Research Methodology

The study employed a mixed research approach (qualitative and quantitative data), semi-structured open-ended and close-ended questionnaires guided by student-based participation. An online questionnaire tool, Google Forms (refer to Appendix 1), has been generated to collect data for 88 students. Open-ended questions were used to collect the qualitative data, and closed-ended questions were intended to gather quantitative data describing the learner's understanding in their

own words. The questions were compiled by using different types of questioning tools such as multiple choice, paragraph response, and rating scale format, as it allowed participants to share their views on how group work strategies have improved the participants' ability to grasp programming concepts. The ethical clearance [FEDREC18-09-23-4] for the study was obtained from the university where this research was conducted. All participants were fully informed about the study, and a consent form for each participant was provided before the questionnaire.

This study proposed using group work strategies and a social constructivism theoretical framework to support students in grasping computer programming. The association of the qualitative data and quantitative data assisted the researcher in understanding the effect of group work strategies on first-year university students. Eighty-eight (88) students enrolled for the CP course in a selected university in the Eastern Cape formed the population of this study.

5. Data Analysis and Discussions

The researcher used a questionnaire with a four-point Likert scale (with no 'neutral' option), which contained multi-item open and close-ended statements. The initial research population consisted of 88 students in the first year and extended programme of a CP course. The following sub-sections consist of the grouped results and explanations of the findings.

5.1 The Role of Group Discussion

Table 1 shows student responses to the premise that group discussion was important in improving their knowledge, skills, and understanding.

Table 1: The role of group discussion and role of mother tongue in discussions

#	Student feedback on group discussions and course engagement	Strongly disagree	Disagree	Strongly agree	Agree
1.1	I understood the course topics better after I discussed them with my group.	2.1%	8.3%	20.8%	68.8%
1.2	Being part of a group that discovered things together about programming helped me to believe that I could become a good programmer.	2.1%	10.4%	27.1%	60.4%
1.3	The environment created by the teacher encouraged us to engage in meaningful discussion.	0%	2.1%	50%	47.9%
1.4	Group discussions in my mother tongue helped me to understand the course content.	6.2%	14.6%	27.1%	52.1%
1.5	Being able to discuss course content with other students who understood the work better than me, helped me to write better programs.	0%	8.3%	35.4%	56.3%

As demonstrated in Table 1, there is an agreement with the principle and a strong correlation. Respondents said they did their homework to participate and learn in classroom discussions. The context around understanding the course topic is approximately 69% strongly agree, and 21% agree, giving me a total of 90%. Student responses to the premise that mother-tongue discussions have a combined 88% strongly agree and agree, which play an important role in improving learning. The results are aligned with the study by Bisai and Singh (2019) on the effective use of translanguaging in promoting collaborative learning. The results are not as strongly positive as the others; this can be explained by the small sample size and the number of students proficient in English. Even so, each item shows agreement with the premise and appears to confirm the validity of the multiliteracies and code-switching perspectives. The results also show the importance of these perspectives in a collaborative learning environment.

5.2 Role of the MKO

The role of the MKO is to provide various levels of support and assistance over iterations of task completion. When learners have internalized the strategies and language for completing the task, they can do it without help. Figure 2 visually represents survey results concerning the impact of MKOs in a learning environment, specifically within a course on development software. MKOs refer to individuals with a higher level of knowledge about a particular topic, including lecturers and peers.

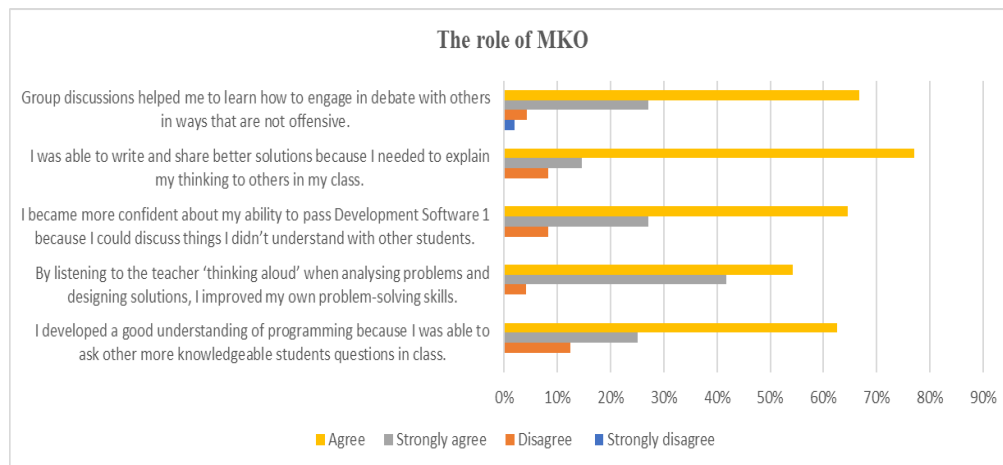


Figure 2: Role of the MKO

Figure 2 shows student responses to the premise that interaction with a “more knowledgeable other” was important in improving their learning. Each of the items shows a distinct agreement and strong correlation. Respondents agree that discussions with more knowledgeable peers helped them to understand their mistakes and to write better computer programs. There is productivity from the role of MKO, which encourages students to attend class at all times. Most participants

either agree or strongly agree with this statement, indicating that group discussions facilitated by MKOs effectively taught respectful debating skills. Evans (2020) and Webb et al. (2014) highlighted that structured group discussions could promote social skills, including respectful communication and conflict resolution. Class attendance is important because students can ask questions to get a better understanding. Approximately 92% agreed that sharing solutions to given problems during their group interaction assisted them in grasping the content. In addition, respondents indicated 94% that the role of MKO helped them to learn how to engage in debate with others in unoffensive ways. Allowing them to collaborate and learn from each other's approaches could improve their problem-solving skills. The statement of writing and sharing the solutions received a high level of agreement, with most participants acknowledging that explaining concepts to others enhanced their clarity and problem-solving capabilities (Jung et al., 2024). Forslund Frykedal and Hammar Chiriak (2018) argued that explaining solutions to peers can deepen understanding and facilitate better cognitive integration of new knowledge. There is significant agreement on this point, suggesting peer discussions increase confidence by clarifying doubts and enhancing comprehension. Ryan and Deci (2017) noted that academic self-efficacy can be boosted through collaborative learning environments where students feel supported by peers. Social constructivism, which emphasizes learning through social interaction and collaboration, works effectively for group work in programming education by allowing students to share diverse perspectives, solve problems collaboratively, and construct knowledge together (Vygotsky, 1978).

The responses overwhelmingly suggested that MKOs, whether students or lecturers, play a crucial role in enhancing learning outcomes in software development education. This is evident in the development of problem-solving skills and programming proficiency. The positive feedback across all statements underlines the importance of interactive and supportive learning environments that leverage the knowledge and skills of more experienced individuals. This approach not only assists the understanding of complex concepts but also cultivates confidence and communicative competence among learners. It aligns with educational research that advocates for collaborative and scaffolded learning approaches.

5.3 Improvement in Social and Communication Skills

Figure 3 presents responses from participants regarding the impact of group discussions and participation in class on their social and communication skills. It shows overwhelming agreement that social constructivism helped improve social and communication skills.

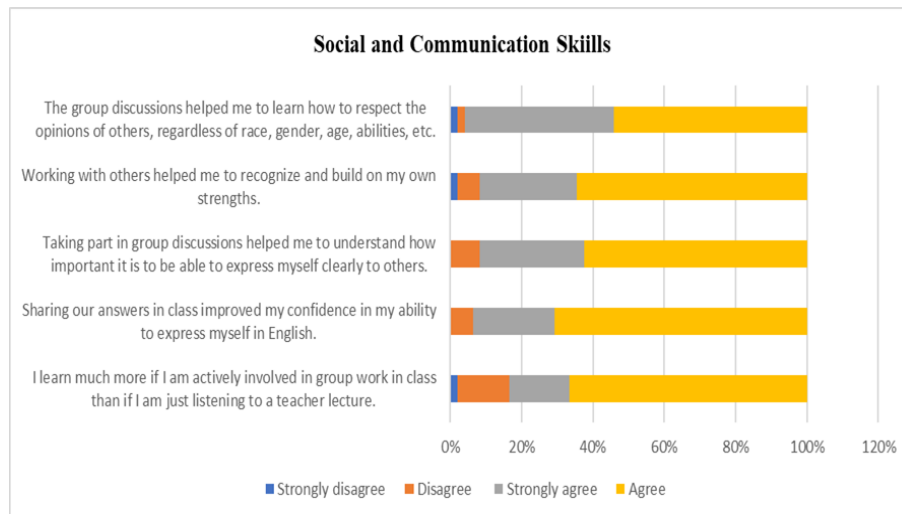


Figure 3: Social and communication skills

As is evident from the data presented in Figure 3, approximately 98% of the respondents agreed that group discussions helped them learn how to respect the opinions of others. The results showed a strong consensus that social constructivism helped improve self-efficacy. Students strongly indicated (98%) that they became more confident about their abilities to pass the course with the help of group discussions during class and that they could express themselves freely in English. Most participants agreed or strongly agreed with these statements, indicating that group discussions effectively enhanced respect for diverse opinions. Many participants agree or strongly agree, suggesting that group work helped individuals identify and develop their strengths. A substantial majority agree or strongly agree, indicating that active participation in class discussions boosted their confidence in using English. A very high level of agreement shows a strong preference for interactive and participative learning methods over traditional lectures.

Overall, the graph shows that the participants perceive group discussions and interactive class activities positively, contributing significantly to their social and communication skills. This aligns with educational research suggesting active learning environments can enhance communication skills and self-confidence (Freeman et al., 2014; Nguyen et al., 2021). Furthermore, it supports the idea that collaborative learning improves academic achievement and develops essential interpersonal skills (Wei et al., 2022; Yilmaz & Yilmaz, 2023).

5.4 Other Findings Based on Open-Ended Questions

To examine the student's ability to understand the concept using group work strategies, the open-ended questions were posed and the following are the few responses. Apart from the close-ended findings, the lecturer observed overwhelming interactivity, experience gained, and excitement during practical sessions. The next sections show the positive feedback from the participants.

5.4.1 *Group work strategies helped me to understand the course content*

Out of 42 responses, a few selected responses as shown in the following paragraph:

Participant 1: In groups, we always make sure that no one is left behind

Participant 2: It helped me because we communicated in my mother tongue

Participant 3: We explained solutions to each other

Participant 4: Working in groups allowed me to identify my strengths and capitalise on my weaknesses, and through that becoming a formidable student.

Participant 5: It boosts my confidence to trust my solutions

Participant 6: Improve your ability to understand and academically

Participant 7: In groups, we share different views and ideas about the course which in a way helps you to understand better.

Participant 8: It helped me to understand the course better because we shared our answers and discussed them so that all of us could understand the content.

The statement emphasized the importance of group work strategies in promoting active learning, improving comprehension, and developing student collaboration. These benefits ultimately lead to a more fulfilling educational experience, as seen by the participant's comments. It can be observed that by actively applying course content to everyday scenarios or problem-solving activities within a group setting, students can strengthen their understanding and develop critical thinking skills. This hands-on approach can make the course material more relevant and applicable, promoting a deeper connection with the subject matter. The responses demonstrated that group discussions, explanations, and peer feedback probably assisted them in gaining clarity on the course that they might have found difficult or confusing at first while studying independently.

5.4.2 *The most practical activity section*

Out of 40 responses, a few selected responses as shown in the following paragraph:

Participant 1: Yes, because you will be able to understand the syntax

Participant 2: It does help and make things easier to understand.

Participant 3: Yes. It's more fun and easier to understand in a language most of us use for communication.

Participant 4: Yes, it did, because it's easy to understand the concepts.

Based on the responses from the participants, it can be observed that the majority of those who actively participate in practical activities could say that it is impacting them positively, which is something that will positively impact the outcome or the kind of grade or marks that they are going to have at the end upon completion of the course.

5.4.3 *The experience gained during group discussions*

Out of 39 responses, a few selected responses as shown in the following paragraph:

Participant 1: In groups, we always make sure that everyone involved understands completely

Participant 2: Allow students to group themselves and express their views

Participant 3: Yes, being concise with what is communicated and listening actively to other students are among the skills/experiences gained from group discussions. There are group work strategies in place used in other courses.

Participant 4: Always group members must focus and pay attention to any member and trust each other.

Participant 5: To be comfortable and confident about your answers and learn to listen to each other.

Participant 6: Everyone should contribute. In group discussions there are no right or wrong answers, we learn from each other.

Participant 7: I would encourage each course to have groups of not more than 5 students so that they will work and not play.

It is observable that group work allows students to interact socially and freely with their peers, promoting a sense of teamwork. This social aspect of learning can enhance motivation, confidence, and overall satisfaction with the learning experience. Building relationships with classmates through collaborative activities can also create a supportive learning environment and encourage mutual support and accountability.

5.4.4 The spirit of 'ubuntu' encouraged to develop in the classroom helped them feel like they 'belonged to a community'

Out of 43 responses, a few selected responses as shown in the following paragraph:

Participant 1: Agree, I realized that I'm not different from anyone and they probably make the same mistakes that I do occasionally.

Participant 2: Agree we shared information

Participant 3: Agree because they never do something wrong to me and others

Participant 3: Agree To respect each other as group members and individuals from different cultures and religions.

The responses highlighted the transformative impact of Ubuntu on building a sense of community and belongingness in the classroom and promoting comprehensive, caring, and compassionate relationships among students and lecturers. Based on the responses, students are likelier to demonstrate empathy towards their classmates, offer assistance when required, and celebrate each other's successes. This culture of support creates a safe and nurturing environment where individuals feel encouraged about the chosen course and grow both academically and personally.

5.5 Overall Rating about the Effectiveness of Group Work Strategies

Figure 4 illustrates survey responses from 88 participants on their satisfaction with the effectiveness of group work strategies in the Development Software 1 course. The student's responses make it clear that the group discussions inspired them to study hard and encouraged them to participate in beneficial academic challenges while

keeping friendly ties. The following figure demonstrates the overall rating of the effectiveness of group work strategies during programming class.

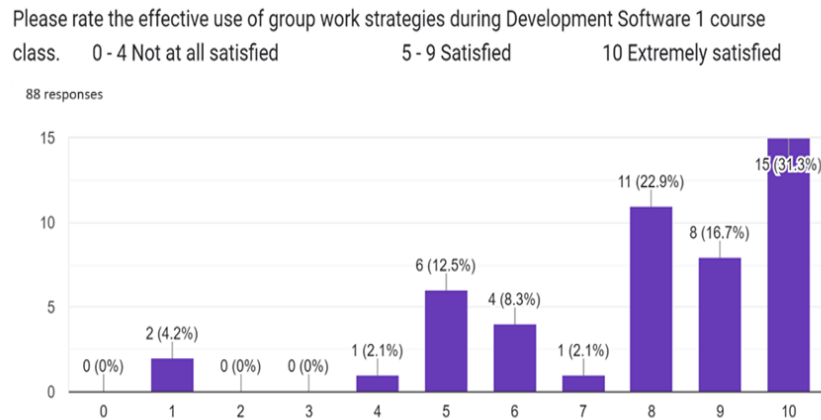


Figure 4: Overall rating of the effectiveness of group work strategies

Figure 4 shows that the overall rating indicates that 93.7% of the respondents are satisfied or extremely satisfied, with satisfied being 62.5% and extremely satisfied being 31.3%. The data show a clear skew towards higher satisfaction levels, particularly noted in the substantial percentages for ratings 8, 9, and 10. This distribution suggests that the majority of students found the group work strategies in the course to be highly effective, enhancing their learning experience significantly. Johnson et al. (2024) state that satisfaction with group work often correlates with improved learning outcomes, as students who are satisfied with their group interactions are more likely to engage deeply with the content. This is evidence that the lecturer uses group work strategies, thus indicating an effective solution for addressing the student's participation during class. Hanks et al. (2011) and Jang et al. (2014) explored the role of group work in software development courses, highlighting that collaboration mirrors professional software development environments, thus providing practical skills alongside academic learning. Laal and Ghodsi (2012) discuss the benefits of collaborative learning, noting that such strategies can enhance problem-solving skills and deepen understanding by enabling students to share diverse perspectives and solutions.

The overwhelmingly positive responses (ratings 8 to 10) align with educational research suggesting that effective group work strategies can significantly enhance learning experiences in technical disciplines like software development. These strategies not only help develop technical skills but also cultivate soft skills such as communication, teamwork, and problem-solving, which are essential in professional environments. The feedback from this survey provides valuable insights for educators in similar courses to refine and emphasize collaborative learning elements to boost student satisfaction and educational outcomes.

In general, based on the results that are presented from the above findings, it clearly shows that the:

- Students gain new skills and knowledge from one another through conversation, which also boosts their confidence and increases motivation.
- Students were driven and self-assured after engaging in this new way of teaching and learning.
- Students valued different methods for problem-solving solutions.
- Social constructivism theory allowed the lecturer to approach the students holistically, encourage creativity, and cater to diversity.
- Improved student performance, especially students from previously disadvantaged rural areas just entering the university for the first time.

6. Conclusion

Firstly, this paper aims to contribute to adopting group work strategies, particularly at the first-year level, to improve student performance in CP. Also, to demonstrate that these strategies support students understanding, making it easier for them to express their understanding in English, the language of learning and teaching. Communication, confidence, and motivation are closely related and contribute to students' ultimate objective of succeeding in their academic courses.

Secondly, students agreed that the group discussions encouraged them to pursue other CP courses in the future in addition to the Introductory course. It was discovered that students grasped the material better and picked up new concepts more quickly when participating more actively in the learning process. Most researchers recommended that one of the skills considered essential in the 21st-century learning environment is CP proficiency. As a result, efforts are required to assist students to strengthen and develop this talent. In this sense, courses in CP could provide students with a useful framework for learning such a crucial ability. This study shows that problem-solving using group work strategies can be implemented by programming lecturers to assist students in enhancing their programming abilities. Undoubtedly, more studies are required to focus on the impact of integrating problem-solving using group strategies on students learning programming in terms of their ability to develop strong programming skills.

7. Limitations, Recommendations, and Future Research

Some limitations of this study should be well-known. The study was conducted within a specific context (at one South African University) for first-year and extended programme students enrolled for CP. The study was also focused on exploring group work strategies to teach the CP course to assist students from disadvantaged backgrounds to grasp programming skills better. It can sometimes be difficult to ensure that all group members contribute equally, leading to scenarios where some students might rely on others to complete programming tasks, thereby not learning the material themselves. Furthermore, different work styles, conflicting commitment levels, and interpersonal dynamics can all impede productive teamwork and the

educational process. Dominant personalities may overshadow quieter students, which hinders equitable participation and educational opportunities. Resource intensity in a case whereby group work often requires more resources, such as classroom space, instructional support, and technological tools, which may not be readily available in all educational settings.

Clear objectives and roles for both lecturers and students are recommended when implementing group work strategies in CP. Diverse skill sets among group members are also encouraged to foster collaborative learning, as are effective communication and conflict-resolution techniques. Peer evaluation mechanisms are integrated to ensure accountability and fairness, and lecturers and students can benefit from training and support to increase the effectiveness of group work activities. While rural schools should create conducive environments with enough space and technology to support collaborative programming projects, curriculum designers should incorporate structured group work tasks that align with learning outcomes and provide resources for assessing group contributions. The subsequent portion of the study would concentrate on creating a teaching and learning model based on experts' opinions that could be used as a guide to improve student's programming skills and, thus, the future direction. It is strongly recommended that academics should passionately pursue transformation and redress in higher education. Social constructivism offers valuable tools for achieving this goal, ensuring graduates are well-prepared with essential programming skills.

7. References

- Aissa, M., Al-Kalbani, M., Al-Hatali, S., & BinTouq, A. (2020). Novice learning programming languages in Omani higher education institution (Nizwa University) issues, challenges and solutions. In A. Al-Masri, & Y. Al-Assaf (Eds.), *Sustainable development and social responsibility* (Vol. 2; pp. 143-148). Advances in Science, Technology and Innovation. Springer Nature. https://doi.org/10.1007/978-3-030-32902-0_18
- Bisai, S., & Singh, S. (2019). Bridging the divide: Collaborative learning and translanguaging in multilingual classrooms. *Fortell—A Journal of Teaching English Language and Literature*, 39(2), 46-57. https://www.researchgate.net/publication/334784891_Bridging_the_Divide_Collaborative_Learning_and_Translanguaging_in_Multilingual_Classrooms
- Biswas, S. (2023). Role of ChatGPT in computer programming. *Mesopotamian Journal of Computer Science*, 2023, 9-15. <https://doi.org/10.58496/MJCSC/2023/002>
- Bosse, Y., & Gerosa, M. A. (2017). Why is programming so difficult to learn? Patterns of difficulties related to programming learning mid-stage. *ACM SIGSOFT Software Engineering Notes*, 41(6), 1-6. <https://doi.org/10.1145/3011286.3011301>
- Bravo-Sotelo, K. P. (2020). Exploring the Tagalog-English code-switching types used for mathematics classroom instruction. *IAFOR Journal of Education*, 8(1), 47-64. <https://files.eric.ed.gov/fulltext/EJ1245827.pdf>
- Bygate, M. (1987). *Speaking*. Oxford University Press.
- Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12(2), ep272. <https://doi.org/10.30935/cedtech/8247>

- Chetty, J. & van der Westhuizen, D. (2017). The use of learning tools for active knowledge construction to develop coding skills. In J. Johnston (Ed.), *Proceedings of EdMedia 2017* (pp. 718-722). Association for the Advancement of Computing in Education (AACE). <https://www.learntechlib.org/p/178380>
- Chetty, J., & van der Westhuizen, D. (2015). *Towards a pedagogical design for teaching novice programmers: Design-based research as an empirical determinant for success* [Conference session]. Koli Calling '15: Proceedings of the 15th Koli Calling Conference on Computing Education Research (pp. 5-12). <https://doi.org/10.1145/2828959.2828976>
- Costa, E. B., Fonseca, B., Santana, M. A., de Araújo, F. F., & Rego, J. (2017). Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Computers in human behavior*, 73, 247-256. <https://doi.org/10.1016/j.chb.2017.01.047>
- Dörnyei, Z., & Thurrell, S. (1994). Teaching conversational skills intensively: Course content and rationale. *ELT Journal*, 48(1), 40-49. <https://doi.org/10.1093/elt/48.1.40>
- Evans, C. (2020). *Measuring student success skills: A review of the literature on critical thinking*. National Center for the Improvement of Educational Assessment. <https://files.eric.ed.gov/fulltext/ED607780.pdf>
- Forslund Frykedal, K., & Hammar Chiriach, E. (2018). Student collaboration in group work: Inclusion as participation. *International Journal of Disability, Development and Education*, 65(2), 183-198. <https://doi.org/10.1080/1034912X.2017.1363381>
- Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences of the United States of America*, 111(23), 8410-8415. <https://doi.org/10.1073/pnas.1319030111>
- Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: A literature review. *Computer Science Education*, 21(2), 135-173. <https://doi.org/10.1080/08993408.2011.579808>
- Ideris, N., Baharudin, S. M., & Hamzah, N. (2019, March). *The effectiveness of scratch in collaborative learning on higher-order thinking skills in programming subject among year-six students* [Conference session]. 4th ASEAN Conference on Psychology, Counselling, and Humanities (ACPCH 2018) (pp. 421-425). Atlantis Press. <https://doi.org/10.2991/acpch-18.2019.99>
- Jang, W., Gao, H., Michaeli, T., & Kasneci, E. (2024, June). *Exploring communication dynamics: Eye-tracking analysis in pair programming of computer science education* [Conference session]. Proceedings of the 2024 Symposium on Eye Tracking Research and Applications (pp. 1-7). <https://doi.org/10.1145/3649902.3653942>
- Johnson, D. W., Johnson, R. T., & Smith, K. A. (2024). Cooperative learning: Improving university instruction by basing practice on validated theory. *Journal on Excellence in College Teaching*, 25(3&4), 85-118.
- Jung, J., Shin, Y., Chung, H., & Fanguy, M. (2024). The effects of pre-training types on cognitive load, self-efficacy, and problem-solving in computer programming. *Journal of Computing in Higher Education*, 1-22. <https://doi.org/10.1007/s12528-024-09407-3>
- Khomokhoana, P. J. (2023). Understanding elements, strengths and challenges of explicit instruction for the teaching of computer programming (to post-graduate students). *The Independent Journal of Teaching and Learning*, 18(1), 59-80. https://scielo.org.za/scielo.php?script=sci_arttext&pid=S2519-56702023000100005

- Kovari, A., & Katona, J. (2023). Effect of software development course on programming self-efficacy. *Education and Information Technologies*, 28(9), 10937–10963. <https://doi.org/10.1007/s10639-023-116178>
- Laal, M., & Ghodsi, S. M. (2012). Benefits of collaborative learning. *Procedia – Social and Behavioral Sciences*, 31, 486–490. <https://doi.org/10.1016/j.sbspro.2011.12.091>
- Liu, X., Li, L., & Zhang, Z. (2018). Small group discussion as a key component in online assessment training for enhanced student learning in web-based peer assessment. *Assessment & Evaluation in Higher Education*, 43(2), 207–222. <https://doi.org/10.1080/02602938.2017.1324018>
- Lukose, J. M. (2021). Effects of guided inquiry-based learning and teaching approach on students' confidence, motivation and communication skills in an introductory computer programming course. *Multicultural Education*, 7(12), 20–33. <https://doi.org/10.5281/zenodo.5756487>
- Nair, P. R. (2020). Increasing employability of Indian engineering graduates through experiential learning programs and competitive programming: Case study. *Procedia Computer Science*, 172, 831–837. <https://doi.org/10.1016/j.procs.2020.05.119>
- Ng, O. L., Leung, A., & Ye, H. (2023). Exploring computational thinking as a boundary object between mathematics and computer programming for STEM teaching and learning. *ZDM Mathematics Education*, 55(7), 1315–1329. <https://doi.org/10.1007/s11858-023-01509-z>
- Nguyen, K. A., Borrego, M., Finelli, C. J., DeMonbrun, M., Crockett, C., Tharayil, S., & Rosenberg, R. (2021). Instructor strategies to aid implementation of active learning: A systematic literature review. *International Journal of STEM Education*, 8, 1–18. <https://doi.org/10.1186/s40594-021-00270-7>
- Nisan, N., & Schocken, S. (2021). *The elements of computing systems: Building a modern computer from first principles*. MIT press.
- Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*, 11(1), 1–17. <https://doi.org/10.1080/20004508.2019.1627844>
- Piwek, P., & Savage, S. (2020, February). *Challenges with learning to program and problem solve: An analysis of student online discussions* [Symposium]. Proceedings of the 51st ACM Technical Symposium on Computer Science Education (pp. 494–499). <https://doi.org/10.1145/3328778.3366838>
- Robins, A. V. (2019). 12 – Novice programmers and introductory programming. In S. A. Fincher, & A. V. Robins (Eds.), *The Cambridge handbook of computing education research* (pp. 327–376). Cambridge University Press.
- Rosenberg-Kima, R. B., Merrill, M. D., Baylor, A. L., & Johnson, T. E. (2022). Explicit instruction in the context of whole-tasks: The effectiveness of the task-centered instructional strategy in computer science education. *Educational Technology Research and Development*, 70(5), 1627–1655. <https://doi.org/10.1007/s11423-022-10143-7>
- Susanti, W. (2021). An overview of the teaching and learning process basic programming in algorithm and programming courses. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(2), 2934–2944. <https://turcomat.org/index.php/turkbilmat/article/view/2332>
- Ryan, R. M., & Deci, E. L. (2017). *Self-determination theory: Basic psychological needs in motivation, development, and wellness*. Guilford publications.
- Ubaidullah, N. H., Mohamed, Z., Hamid, J., Sulaiman, S., & Yussof, R. L. (2021). Improving novice students' computational thinking skills by problem-solving and metacognitive

- techniques. *International Journal of Learning, Teaching and Educational Research*, 20(6), 88–108. <https://doi.org/10.26803/ijlter.20.6.5>
- Vygotsky, L. S. (1978). *Mind in society: Development of higher psychological processes*. Harvard University Press.
- Webb, N. M., Franke, M. L., Ing, M., Wong, J., Fernandez, C. H., Shin, N., & Turrou, A. C. (2014). Engaging with others' mathematical ideas: Interrelationships among student participation, teachers' instructional practices, and learning. *International Journal of Educational Research*, 63, 79–93. <https://doi.org/10.1016/j.ijer.2013.02.001>
- Wei, Y., Shi, Y., MacLeod, J., & Yang, H. H. (2022). Exploring the factors that influence college students' academic self-efficacy in blended learning: A study from the personal, interpersonal, and environmental perspectives. *SAGE Open*, 12(2), Article 21582440221104815. <https://doi.org/10.1177/21582440221104815>
- Yilmaz, R., & Yilmaz, F. G. K. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence*, 4, Article 100147. <https://doi.org/10.1016/j.caeai.2023.100147>
- Zhang, X., Zhang, C., Stafford, T. F., & Zhang, P. (2013). Teaching introductory programming to IS students: The impact of teaching approaches on learning performance. *Journal of Information Systems Education*, 24(2), 147–155. <http://jise.org/Volume24/n2/JISEv24n2p147.pdf>

Appendix 1

The following link was used for semi-structured open-ended and closed-ended questionnaires, and the responses were retrieved.

https://docs.google.com/forms/d/1TQewyebTBYtUiLoF7IymTB9rSqZgJSQ_u_bZpsAoVWo/edit