SMAC-Based Programming Tool: Validating a Novel System Architecture

https://doi.org/10.3991/ijet.v17i13.29337

Salihin Rahim(✉), Saiful Omar, Thien Wan Au, Irwan Mashadi Mashud
Universiti Teknologi Brunei, Gadong, Brunei Darussalam
p20190006@student.utb.edu.bn

Abstract—Although SMAC (Social, Mobile, Analytics, Cloud) allows pedagogical innovation due to its ability in expanding communication during the learning process, its exploitation for programming learning has yet to occur. Therefore, this study proposes a novel system architecture for a SMAC-based programming learning tool designed to enhance programming students' problem-solving and collaborative skills. In evaluating the effectiveness of this architecture, this study conducted multiple two-week controlled experiments involving 71 introductory programming students. The experiment involves administering pre-and post-study surveys and tests to measure the enhancement in the two skills. The overall findings of this study indicated a statistically significant improvement in both the student's problem-solving ability and collaborative skills. Most importantly, these findings suggest that the synergy of all four SMAC elements in a single programming learning tool has substantial benefits to programming learning.

Keywords—SMAC (Social Mobile Analytics Cloud), programming, collaboration, problem-solving, distant learning

1 Introduction

Studies have proven that students have difficulties learning computer programming [1], often argued due to the lack of problem-solving ability among the students. Such a lack makes them unable to think algorithmically [2], especially in solving programming problems. However, this problem remains unresolved despite being researched in many studies, suggesting its research immaturity. It is vital to handle this problem as it carries several unintended negative impacts on students, such as losing motivation after repeatedly making mistakes in their programming activities [3]. Consequently, this problem might increase the failure rate in programming courses.

Additionally, the current approach to teaching computer programming focuses too much on teaching programming syntaxes and semantics rather than allowing the students to develop their problem-solving skills [4]. Students have no trouble memorising programming syntaxes and understanding their semantics [5]. However, they are often unable to use their programming skills to plan and structure a solution to a programming problem. Hence, developing students' problem-solving abilities is essential before teaching them programming vocabulary [6].

Therefore, this study foresees the importance of the current teaching and learning approach to innovating. For years, the programming learning domain has seen many innovative learning tools to mitigate this problem. However, still, there is no silver bullet learning tool that could solve all issues related to problem-solving skills. A systematic literature review (SLR) by [7] provided an exciting discovery when no study tried to adopt SMAC (Social, Mobile, Analytics, Cloud) into the programming domain, although this concept is not new anymore. SMAC arose from the proliferation and advancement of four disruptive technologies: social media, analytical technologies, mobile computing, and cloud computing [8]. This concept is often seen as a new pedagogical innovation enabler due to its ability in expanding communication across class boundaries [9], [10].

This study was conducted to extend the work of [7] by investigating the synergic impacts of all four SMAC technologies on the programming learning domain. This article reports a novel system architecture for a SMAC-based programming learning tool (SPLT) that combines features from varieties of provenly-effective state-of-the-art learning tools related to SMAC. Most importantly, this study aims to evaluate the extent to which SPLT enhances the problem-solving ability and collaborative skills of programming students. To this end, a controlled experiment involving 71 participants was conducted to answer the following research questions:

1. Does SPLT impact the problem-solving abilities of the participants?
2. Does SPLT impact the collaborative skills of the participants?

2 Literature reviews

2.1 SMAC elements in programming learning tools

SMAC technology is a key enabler for more innovative learning pedagogies [9], [10] that extend learning and student-educator interactions beyond the class boundaries. However, existing studies have yet to exploit the synergy of all four SMAC technologies (elements) for programming learning, as reported in [7]. Besides, [11] also urges the need for more development of theories and models of SMAC-based pedagogy.

Moreover, [7] reported multiple roles of SMAC technologies in aiding programming learning. For example, [7] discovered that social technologies are often used to enhance in-class collaboration, and mobile technologies were often used to enable a mobile-learning environment and for a development host. Additionally, analytical techniques are increasingly applied in programming learning by analysing students' behaviour during programming activities or making early predictions through machine learning. Meanwhile, existing studies used cloud technologies to establish a cloud-based programming environment that allows students to perform programming activities on the cloud.

Although meta-analysis is not possible in [7] due to the heterogeneity of the included papers, the study identified the benefits of using a programming learning tool that combines SMAC technologies to enhance students' performance. For instance, [12] combined social and analytical technologies to develop a new collaborative programming learning environment and effectively group students. Their study concluded that students improved their motivation to learn, programming understanding and communication skills after using their system.

Meanwhile, [13], [14] examined the effects of using mobile devices and social networking sites in aiding programming learning and discovered positive enhancements in students' abstract thinking and students' interactions.

The studies above demonstrate the benefits of combining SMAC technologies in programming learning. Therefore, this study postulates that the synergy of all four SMAC technologies would provide better outcomes for programming students. However, to the best of our knowledge and the findings in [7], there is no framework or pedagogical model for exploiting such synergy.

2.2 Problem-solving skills

Problem-solving is a technique requiring the transfer, adaption, and application of the knowledge they have learned to a new, unfamiliar situation [15]. Problem-solving or critical thinking is essential regardless of whether education is online or physical [16]. Problem-solving skills are an integral part of programming learning [18] and a factor in excelling in the course [6]. A good problem-solver possesses two abilities: first, they possess background programming knowledge learned through programming exercises. Second, they can apply their background knowledge to a new programming exercise. However, recent investigations have demonstrated that programming students put less effort into developing their problem-solving skills [6].

Mhashi and Alakeel [5] claimed that students are often unable to use their programming skills to plan and structure a solution to a programming problem. Researchers concluded that mastering the syntaxes alone is insufficient without possessing good problem-solving skills [2], [14], [17]–[20]. According to [22], developing problem-solving skills is more important than learning multiple programming languages because students' inability to think algorithmically would restrict their knowledge transfer. Hence, educators must cultivate the students' problem-solving skills by focusing more on improving their algorithmic thinking skills.

2.3 Collaborative skills

Collaboration is essential for ensuring impactful programming learning [22], [23]. Chorfi et al. [23] argued that learning programming collaboratively requires students to work together, share skills, and learn from one another. These activities help them learn programming better as they help build a stronger problem-solving ability [22]. However, students will not collaborate naturally in a group [24]. Hence, many studies have provided various guidelines to foster student collaboration [25]. For example, [26] discovered 19 collaborative resources from existing studies to foster student collaboration during programming learning. [7], [26] discovered the two most frequently used collaborative formats in programming learning: 1) pair programming and 2) group programming (more than two students).

Pair programming pairs two students together while solving a programming problem using one computer, and each student takes one role at a time, either as a driver or a navigator [27]. Studies documented its effectiveness in improving students' collaborative skills [27]–[30]. However, existing studies usually focused on investigating the impacts of in-class pair programming and lesser on distributed one [7].

Furthermore, communication is an essential aspect of successful collaborative learning [26]. A chatting tool is the most frequently used communication medium, as reported in [26], making collaboration possible even with geographically distributed students. For example, [23], [30] used a chatting tool to support synchronous collaboration between programming students and reported a positive result. Additionally, using chat tools in class encouraged students to communicate their thoughts and problems [12], demonstrating the effectiveness of the chat system to increase collaboration regardless of the students' location.

3 SMAC-based programming learning tool architecture

Figure 1 shows the system architecture for SPLT, which serves as a blueprint for elaborating the adoption of the four SMAC elements into the programming learning environment. It specifies the functions of each SMAC element and how they should work together to establish a functional SMAC-based learning environment. Besides, the architecture describes the data generated by each SMAC element and how other elements can use the data to benefit from the four SMAC elements' synergy.

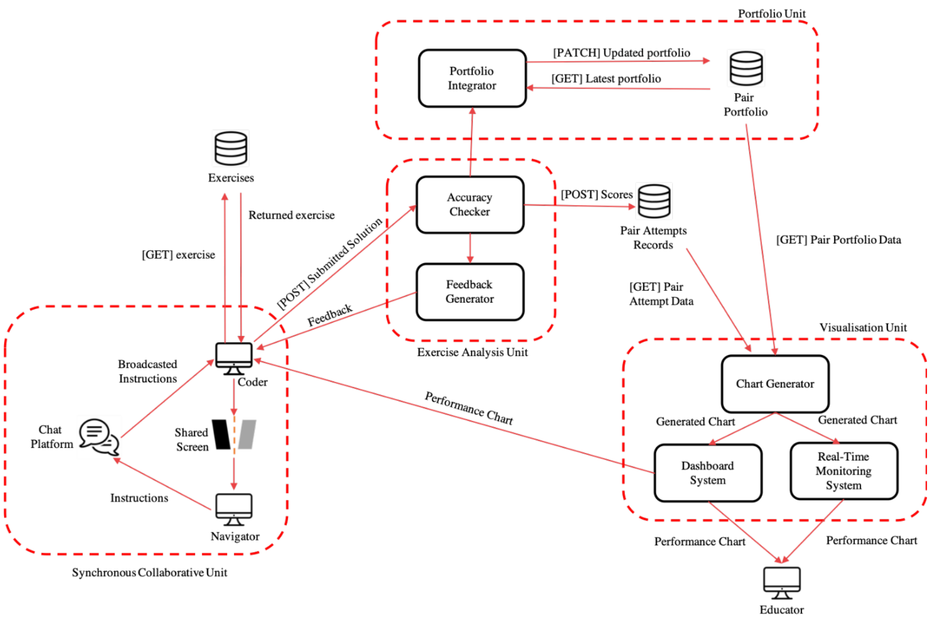


Fig. 1. The header image of online-journals.org

The architecture holds four functional units, namely (1) *Synchronous Collaborative Unit*, (2) *Exercise Analysis Unit*, (3) *Portfolio Unit*, and (4) *Visualisation Unit*. These four functional units work together with twofold aims. Firstly, they aim to establish a cloud-based, collaborative, and analytical-powered learning platform that allows geographically distributed programming students to learn computer programming together. Secondly, they aim to enhance the students' collaborative skills and problem-solving ability by collaborating to solve programming exercises.

3.1 Synchronous collaborative unit

This functional unit combines three SMAC elements (social, mobile, analytics and cloud elements) to create an online programming learning platform with synchronous collaboration capabilities. This unit acts as a social media in this architecture to provide a learning environment where programming students can exchange information [31] and partner with other students in the system. In addition, this unit interacts the most with the students because it provides all the features that help develop their problem-solving ability and collaborative skills.

Studies indicate that pair programming is the most commonly used and provenly-effective collaboration method [27], [30]. Thus, this study proposes the adoption of distributed pair programming into the architecture so that the collaboration can go beyond the geographical boundary.

The architecture proposes combining the benefits of social and cloud elements to enable an online-based synchronous collaboration. One way is by establishing a synchronous code playground (editor), shared in real-time and cloud-based. This social platform allows distant students to collaborate and solve programming exercises while developing their problem-solving skills. Students must be able to view the same editor on their screens. Thus, its interface should be updated synchronously with each student's action.

Furthermore, students who communicate more during a collaborative activity produce better problem-solving outcomes. Besides, an interactive eLearning system could motivate students to learn more [32]. An existing study showed that textual communication increases dialogue, interaction and is helpful for students in an online environment [33]. Thus, this study proposes embedding a synchronous chat tool in the architecture.

Meanwhile, a mobile learning environment is adapted to incorporate SMAC's mobile element. According to [34], ubiquitous and portability are two of the seven core characteristics of a thriving mobile learning environment. Thus, SPLT must be a mobile learning tool that adapts to various screen sizes. Such capability will expand the accessibility of SPLT's content to mobile devices, making learning ubiquitous.

3.2 Exercise analysis unit

This functional unit incorporates only the analytics element into the architecture by holding two main components: (1) Accuracy Checker (AC) and (2) Feedback Generator (FG). These two components perform the system's analytical tasks, aiming to measure students' performance based on their activities in the system. This study proposes using two outcome measures proposed by [35], namely *"success on task"* and *"time on task"*.

There are three factors to determine success in performing a task [33]. The first factor is determining the goal state (GS) that students must achieve to succeed in solving a task. The GS in this study is students' ability to submit a solution that satisfies a model solution. Thus, SPLT must include a platform where educators can submit model solutions for grading.

The second factor is defining a method to determine whether a student reaches a GS. For example, a system could be programmed to automatically detect an event where a student reaches a GS [33]. One method is to use an algorithm that compares a student's solution to a model solution and grades it accordingly.

The third factor is defining a method that unambiguously defines the GS and tells students how to achieve the goal. For this factor, SPLT must clearly explain every programming exercise. For instance, [6] provided textual instructions on their exercises page telling students what they must do to succeed.

Meanwhile, measuring the *"time on task"* requires defining the starting and ending points [33]. A timer can be embedded at the system backend to automatically detect how long students take to solve an exercise and submit their solution.

The AC can perform the first two factors discussed above in the architecture. Following student submissions, the AC will send the calculated score and solving time to the server for database storage. Then, sequentially, the AC will send the calculated score to the FG to generate feedback informing the students of their performance. This study also suggests adopting the *"formative assessment feedback"* technique discussed in [36] to generate the feedback. Formative assessment is a technique that provides feedback to the students instantly as they complete a task to make appropriate adjustments to their learning [37]. Instilling formative feedback in students' daily learning would entice them to learn more, manage it better, and overcome problems in learning better [37].

3.3 Portfolio unit

This functional unit also incorporates only the analytics element into the architecture, but it is responsible for classifying the students based on their performance over time. All the data generated by the students in the system can be used to create a portfolio system (PS). The PS would help educators identify at-risk students quickly and effectively for responsive assistance. Studies have shown the importance of providing immediate assistance [38]–[40].

Rahim et al. [7] identified ways to develop a PS. For instance, [41] used the data collected after students used their online judge's system in the first two weeks of the course and concluded that predictive modelling was helpful for this purpose. However, developing a significant prediction model depends on the quality and volume of the data used. Besides, developing a good prediction model requires mastering different skillsets [41].

Alternatively, the student-generated data in the Synchronous Collaborative Unit (SCU) can be used to generate the PS. Thus, the Portfolio Unit can use students' solution scores to calculate the average scores and use them to classify the students. This study suggests using the classification strategy proposed in [43], as shown in Table 1.

Table 1. Malik and Coldwell-Neilson's [43] classification of programming students based on their average score

Average Score Range	Classification Group
≤ 50	Fail
50 ≤ 64	Low-performing
65 ≤ 84	Medium-performing
85 ≤ 100	High-performing

3.4 Visualisation unit

This functional unit incorporates only the analytics element and holds three main components: *Dashboard System, Real-Monitoring System,* and *Chart Generator*. These components receive data from the Portfolio Unit (PU) and the Exercise Analysis Unit (EAU) and visualise their analytical processes outputs into charts.

The *Dashboard System* possesses two main functions. First, the dashboard must inform students' current performance by instilling formative assessment feedback techniques [36]. Seanosky et al. [34] argued that learning progression charts help students improve their grades. Second, it must provide students with quick access to suitable learning materials based on weak programming topics [44]. Also, future developers could design the dashboard to show the learning progression charts as soon as students log into SPLT.

Meanwhile, the *Real-Time Monitoring System* must support educators to identify at-risk students instantly. One method involves displaying all students' learning progression on a single dedicated page. Additionally, the system must also allow educators to monitor students' progress. These features will help educators keep track of students' progression and provide necessary assistance quickly [45].

Finally, the *Chart Generator* is responsible for generating the analytical charts. This component interacts with the PU to obtain the learning progress data and the EAU for the students' activities data. Most importantly, this study does not restrict the attributes of each chart. For example, the chart can display the calculated score against the number of attempts (score vs attempts) or display solving time against the number of attempts (solving time vs attempts). We postulate that these two attributes will effectively provide formative assessment knowledge to the students.

4 Evaluation methodology

4.1 Participants

This study used convenience sampling to recruit 71 fundamental programming students majoring in computing courses from five institutions in Brunei Darussalam. All institutions provided their ethical clearance to experiment, and all participants provided their consent. The participants' average age was 21.6 years old, of which 42 (59.2%) were males, and 29 (40.8%) were females. Regarding their programming background, the participants have an average of 3.09 years of programming, enrolled in at least four programming modules, and were involved in at least three programming projects in groups. Furthermore, the programming modules in Brunei's institutions include similar programming topics in their fundamental programming modules, including arrays, iterations, and conditions. Hence, the participants had covered similar programming topics regardless of their institutions.

4.2 Procedures

In this experiment, all participants were required to use SPLT and apply conventional learning (CL) to avoid benefiting only one group of the participants. Therefore, this study adopted an AB/BA crossover design with two periods and two sequences. The first group used SPLT in the first period and CL in the second period, whereas the second group used CL in the first period and SPLT in the second period. The experiment was conducted online to simulate a true online learning environment. Each session lasted three and a half hours, so the second period was conducted three days apart. A priori, the carryover factor was considered in this study design as the participants might carry over something from the first period.

Before starting the experiment, the participants were briefed on the purpose, activities, and expected outcomes. Then, all participants signed a consent form and completed a pre-study survey online before being randomly allocated into two groups. The pre-study survey aimed to gather participants' programming backgrounds and perceived collaborative skills. Meanwhile, the pre-study test measured the participants' prior problem-solving skills.

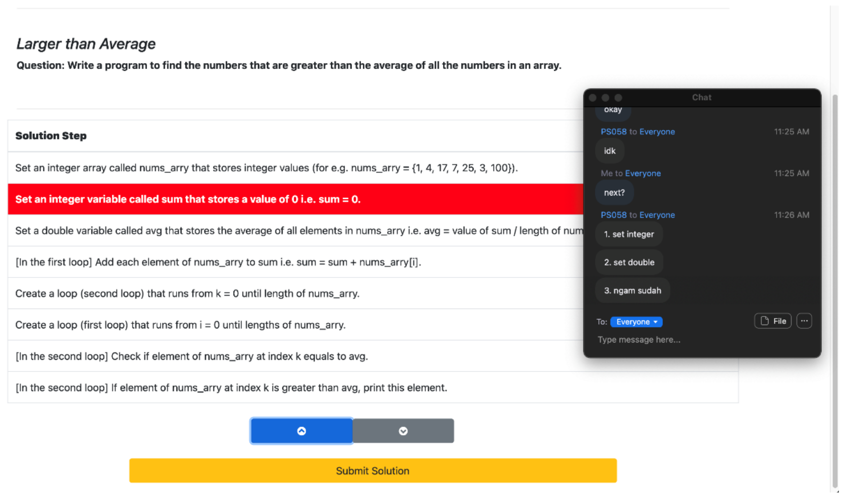
All programming exercises used in this experiment was designed to develop the participants' problem-solving skill, especially in code debugging, tracing, logical thinking, conditional, and iterations. During the experiment, those who used SPLT were given access to the system so they could solve the exercises through the system. Additionally, every two participants who used SPLT were paired based on their pre-study test scores to see the impacts of homogenous pairing. Finally, all pairs used a chat tool to communicate during the experiment. In contrast, those who used CL were not allowed to use SPLT to solve the programming problems, and they had to solve them individually.

In each period, participants from both groups were instructed to solve four programming exercises using their respective techniques within the allotted time. A post-study-test was administered at the end of each period. After the experiment ended, a post-study survey was also administered to gather the participants' perceptions of SPLT.

4.3 Instruments

SPLT proof-of-concept system. For evaluating the architecture, this study developed a proof-of-concept system of SPLT that simulates a cloud-based, analytically-driven, and collaborative programming learning environment. This system supported distributed pair programming by allowing two distant learners to view the same code playground via a web browser (see Figure 2). A chat tool was also embedded to support synchronous communication between the pairs. Furthermore, this study adopted a problem-solving development strategy proposed by [6], which allowed students to solve programming exercises in three ways: (1) rearranging pseudocode in their correct sequence, (2) rearranging programming codes in their correct se1quence, and (3) translating the pseudocode algorithm into programming codes of their preferred language. We believe these three ways of programming learning will induce students' motivation and interest to learn programming, especially when [46] found a relation between task attractiveness and the system's perceived usefulness.

The system also logged students' scores on every programming exercise they attempted. These scores were coupled with exercise ID, time taken to solve the exercise, and the number of actions done to solve the exercise. The data coupling helped the system analyse students' performance on specific programming topics and recommended relevant exercises.



Larger than Average

Question: Write a program to find the numbers that are greater than the average of all the numbers in an array.

Solution Step
Set an integer array called nums_arry that stores integer values (for e.g. nums_arry = {1, 4, 17, 7, 25, 3, 100}).
Set an integer variable called sum that stores a value of 0 i.e. sum = 0.
Set a double variable called avg that stores the average of all elements in nums_arry i.e. avg = value of sum / length of num
[In the first loop] Add each element of nums_arry to sum i.e. sum = sum + nums_arry[i].
Create a loop (second loop) that runs from k = 0 until length of nums_arry.
Create a loop (first loop) that runs from i = 0 until lengths of nums_arry.
[In the second loop] Check if element of nums_arry at index k equals to avg.
[In the second loop] If element of nums_arry at index k is greater than avg, print this element.

Chat

okay

PS058 to Everyone 11:25 AM

idk

Me to Everyone 11:25 AM

next?

PS058 to Everyone 11:26 AM

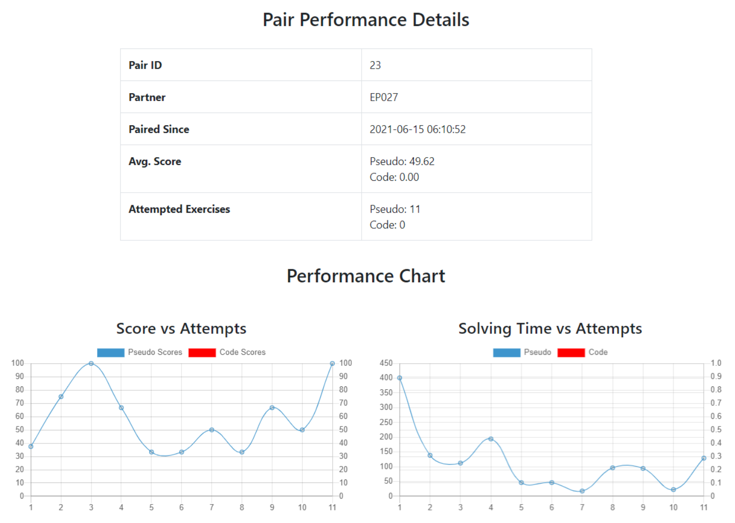
1. set integer

2. set double

3. ngam sudah

Fig. 2. Synchronous and collaborative code playground (adopted from [6])

The system also displayed relevant analytical outputs through multiple line charts for allowing the students to grasp their learning progression. These charts utilised three data generated by the students in the system: (1) score obtained, (2) time taken to solve an exercise, and (3) the number of attempts the students made on a particular exercise.



Pair Performance Details

Pair ID	23
Partner	EP027
Paired Since	2021-06-15 06:10:52
Avg. Score	Pseudo: 49.62 Code: 0.00
Attempted Exercises	Pseudo: 11 Code: 0

Fig. 3. Learning progression chart of paired partners

Collaborative skills measurement. This study adopted a 37-item collaborative skills measurement survey by Tibi [47] to measure participants' collaborative skills enhancement. The survey used a five-point Likert scale with one indicating "strongly disagree", and five indicating "strongly agree". The survey measured five essential elements of effective group collaboration: *positive interdependence, individual accountability, promotive interaction, social skills,* and *group processing*. The survey was administered twice before and after the experiment to measure changes in the perceived collaborative skills.

Problem-solving skills measurement. Participants' problem-solving skills were measured through pre-and post-study tests. The pre-study test was administered before the experiment started, whereas the post-study test was administered after they completed all the exercises given during the experiment. Both tests consisted of five questions that the participants must solve in the allotted time. Additionally, a senior lecturer from Universiti Teknologi Brunei validated the questions and the marking rubric for the tests. Following the first grading, this study appointed a postgraduate student to assess the scores again to avoid biased grading.

Participants SPLT perception measurement. The post-study survey included three open-ended questions that gathered the participants' perspectives on SPLT's ability to develop their problem-solving and collaborative skills and its potential to be used in programming lectures.

5 Results and discussions

This study used IBM SPSS to statistically analyse the differences in the problem-solving and collaborative skills of the participants before and after the experiment. This section discusses the results from all the statistical analyses conducted in this study.

5.1 Differences in problem-solving skills

A linear mixed model test was first conducted to detect any interaction between period and sequence. The result demonstrated that the interaction between period (p = 0.537) and sequence (p = 0.933) were insignificant. Thus, we could safely conclude that the difference in the effectiveness was not due to other variables than the techniques used in the experiment (SPLT and CL). According to the result, SPLT was statistically more effective than CL (F = 17.431, p < 0.01**). The participants' mean scores after using SPLT was 75.23%, while their mean scores after using CL was 61.18%.

Additionally, the post-study test scores in the first period were utilised to investigate any significant improvements in the problem-solving skills from the pre-study test scores. From the paired t-test conducted, both groups had better post-study test scores after the first period. Those who used SPLT in the first period had statistically better post-study score (M = 74.689, SD = 22.96) than their pre-test score (M = 48.63, SD = 15.95, t(35) = 9.609, p < 0.01**, d = 1.601). Similarly, those who used CL also had statistically better post-study scores (M = 60.91, SD = 15.51) than their pre-study test score (M = 48.64, SD = 15.95, t(34) = 4.532, p < 0.01**, d = 0.766). However, an independent t-test on both post-study test scores revealed that participants who used SPLT in the first period had statistically better mean scores than those who used CL (t

= 2.953, p = 0.004, d = 0.701). So, these results interpreted that SPLT had a better ability to improve the participants' problem-solving skills.

Table 2. Results of paired t-test on problem-solving skills after the first period

	N	Pre-test		Post-test		t
		M	***SD***	***M***	***SD***	
Group using SPLT	36	48.10	23.10	74.689	22.96	9.606**
Group using CL	35	48.64	15.95	60.91	15.51	4.532**

Note: ** p < 0.01

A possible explanation for this result might be the pseudocode playground adapted from [6], which effectively builds the participants' algorithmic thinking skills. The participants had to learn problem-solving by first identifying the input, process, and output rather than directly coding the solution. Thus, this finding confirms the pseudocodes' effectiveness to develop problem-solving skills, as claimed in [6].

Moreover, the feedback system might also play a vital role in the positive result. According to [36], students have higher chances of getting good grades if they grasp more knowledge about their learning. However, they discovered that not all students were proactive in viewing their learning charts. Therefore, we took a proactive approach by designing the system to display the learning progression charts as soon as the students logged in and completed an activity, either through recommendations or learning progression charts. The content analysis provided evidence that our proactive design assisted 22% of the participants (*n* = 8) who responded to the survey in understanding their progression and consequently helped them improve their problem-solving skills.

This study also supports the findings of [48], [49] on the effectiveness of homogenously pairing students based on their programming ability. Our content analysis showed that 11% of those who participated in the survey cited team effort as a factor that influences their perception of the efficacy of SPLT in enhancing their problem-solving ability. This finding demonstrates that randomly pairing the participants based on their skills had no detrimental impact on their learning outcomes.

5.2 Differences in collaborative skills

This study conducted a paired samples t-test to distinguish the difference in the participants' perceived collaborative skills before and after using SPLT. The result demonstrated a significant difference in mean perceived collaborative skills scores before (M = 3.838, SD = 0.353) and after using SPLT (M = 4.068, SD = 0.360), t(70) = 5.816, p < 0.01**, d = 0.6907, α = 0.05. The result concludes that there is an advantage in utilising our system for enhancing the perceived collaborative skills of the participants.

Table 3. Results of paired t-test on collaborative skills

	Pre-test		Post-test		t
	M	***SD***	***M***	***SD***	
Collaborative skills enhancement	3.838	0.353	4.068	0.360	5.816**

Note: ** p < 0.01

We believe that the cloud-based and collaborative-enhancing features embedded in SPLT positively impact the participants' collaborative skills. The most prominent feature was the inclusion of functions that support distributed pair programming, which was embedded to represent the social and cloud elements of SMAC. This feature allows the participants to work collaboratively in developing their problem-solving skills [50], where they were required to discuss the solution before submitting it for evaluation.

Consequently, this study indirectly compelled the pairs to interact by providing instruction and feedback, allowing two-sided communication. Rodríguez et al. [28] highlighted the importance of two-sided communication for ensuring a successful collaboration. Additionally, pair programming appoints roles to the participants, which changes after a while. These roles required both participants in a pair to continuously monitor each other's work and provide feedback, including gradually discussing their group performance. As a result, the participants eventually developed their collaborative skills through continuous feedback exchange [47].

Moreover, this study also scripted the participants' collaborative activities [30], including the period to change their roles. Thus, this finding confirms Tsompanoudi et al.'s [30] conclusion that the application of collaboration script influences the success of collaboration in distributed pair programming. Also, the positive result of this study is consistent with prior research on the effectiveness of a chat tool in facilitating student collaboration [12], [29], [30].

6 Conclusions and limitations

This study proposed a novel system architecture for a SMAC-based programming learning tool (SPLT) that combines the state-of-the-art features identified in [7] to enhance students' problem-solving and collaborative skills. This study developed a proof-of-concept system from the architecture and experimented with 71 introductory programming students from five different institutions in Brunei Darussalam with the tool. The overall findings and analysis showed improvements in problem-solving and collaborative skills. These findings suggest that our approach is a promising, supplementary learning tool for programming students to develop these two skills. The findings also indicate that the synergy of all four SMAC elements in a single programming learning tool has substantial benefits to the programming students. Based on these findings, we believe they would help respective educational sectors and future developers comprehend how the SMAC concepts could be used in programming education to enrich students' problem-solving and collaborative skills. Furthermore, the positive findings might entice future developers to use the proposed architecture to develop a more sophisticated and comprehensive SPLT.

However, our current experiment design still has its limitations. Firstly, this study only examined distributed pair programming to enable the social aspect of the system. Future studies might extend the feature by allowing more than two programming students to collaborate in the cloud-based system. Secondly, only a chat tool was used to support the communication between the distant students. We recommend that future studies examine how verbal or video communication could help improve collaboration.

7 Acknowledgements

The authors of this study would like to thank Universiti Teknologi Brunei for financially supporting this study with an Internal Research Grant by reference no: UTB/GSR/2/2020 (3).

8 References

[1] M. S. R. Derus and M. A. Z. Ali, "Difficulties in learning programming: Views of students," *1st Int. Conf. Curr. Issues Educ. (ICCIE 2012)*, no. September 2012, pp. 74–79, 2012. http://dx.doi.org/10.13140/2.1.1055.7441

[2] M. Amaratunga, G. Wickramasinghe, M. Deepal, O. Perera, D. De Silva, and S. Rajapakse, "An Interactive Programming Assistance tool (iPAT) for instructors and novice programmers," in *2013 8th International Conference on Computer Science & Education*, 2013, pp. 680–684. https://doi.org/10.1109/ICCSE.2013.6553995

[3] F. H. Leong, "Automatic detection of frustration of novice programmers from contextual and keystroke logs," in *2015 10th International Conference on Computer Science & Education (ICCSE)*, 2015, pp. 373–377. https://doi.org/10.1109/ICCSE.2015.7250273

[4] A. Gomes and A. Mendes, "A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 2014, pp. 1–8 . https://doi.org/10.1109/FIE.2014.7044086

[5] M. M. Mhashi and A. L. I. M. Alakeel, "Difficulties Facing Students in Learning Computer Programming Skills at Tabuk University," *Recent Adv. Mod. Educ. Technol.*, pp. 15–24, 2013.

[6] S. I. Malik, R. Mathew, R. Al-Nuaimi, A. Al-Sideiri, and J. Coldwell-Neilson, "Learning problem solving skills: Comparison of E-learning and M-learning in an introductory programming course," *Educ. Inf. Technol.*, vol. 24, no. 5, pp. 2779–2796, 2019. https://doi.org/10.1007/s10639-019-09896-1

[7] S. Rahim, S. Omar, T. W. Au, and I. M. Mashud, SMAC (Social, Mobile, Analytics, Cloud)-Based Learning Intervention for Introductory Programming – The Trend in the Past 15 Years, no. March. Springer International Publishing, 2021. https://doi.org/10.1007/978-3-030-68133-3_7

[8] N. Belarbi, N. Chafiq, M. Talbi, and A. Namir, "An Experimental Case Study: Integrating Mobile Dimension from SMAC Technologies (Social, Mobile, Analytics and Cloud) within a SPOC," *Int. J. Sci. Eng. Appl. Sci.*, vol. 3, no. 4, pp. 2395–3470, 2017, [Online]. Available: https://www.researchgate.net/publication/322086662

[9] K. T. Levinsen, "Qualifying online teachers-Communicative skills and their impact on e-learning quality," *Educ. Inf. Technol.*, vol. 12, no. 1, pp. 41–51, 2007. https://doi.org/10.1007/s10639-006-9025-1

[10] E. C. Larson, "Teaching Case Congratulations! ...to the world? One Person's experience with social media," *J. Inf. Syst. Educ.*, vol. 29, no. 3, pp. 127–130, 2018.

[11] H. U. Khan, "The role of SMAC (social media, mobility, analytics, cloud) for students and educators in online education," *J. Theor. Appl. Inf. Technol.*, vol. 98, no. 6, pp. 915–934, 2020.

[12] D. D. Phuong, F. Harada, and H. Shimakawa, "Collaborative Learning Environment to Improve Novice Programmers with Convincing Opinions in Computer Room," in *2009 International Conference on Intelligent Networking and Collaborative Systems*, 2009, pp. 61–66. https://doi.org/10.1109/INCOS.2009.54

[13] M. Maleko, M. Hamilton, and D. D'Souza, "Access to mobile learning for novice programmers via social networking sites," in *2012 7th International Conference on Computer Science & Education (ICCSE)*, 2012, pp. 1533–1538. https://doi.org/10.1109/ICCSE.2012.6295355

[14] A. P. L. Ambrósio and F. M. Costa, "Evaluating the impact of PBL and tablet PCs in an algorithms and computer programming course," in *SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 2010, pp. 495–499. https://doi.org/10.1145/1734263.1734431

[15] J. Carson, "A Problem With Problem Solving: Teaching Thinking Without Teaching Knowledge," *Math. Educ.*, vol. 17, no. 2, pp. 7–14, 2007.

[16] H. Van Der Meijden, N. M. Zaid, and A. H. A. Rashid, "Online Reciprocal Peer Tutoring Approach in Facebook : Measuring Students ' Critical Thinking," *Int. J. Emerg. Technol. Learn.*, vol. 2, pp. 16–28, 2021. https://doi.org/10.3991/ijet.v16i23.27451

[17] I. Chung, C. Chou, C. Hsu, and D. Li, "A programming learning diagnostic system using case-based reasoning method," in *2016 International Conference on System Science and Engineering (ICSSE)*, 2016, pp. 1–4. https://doi.org/10.1109/ICSSE.2016.7551544

[18] D. Diez, P. Diaz, I. Aedo, and C. Fernandez, "DEI-CHECK. Automating the assessment process to improve the informative feedback," in *2008 38th Annual Frontiers in Education Conference*, 2008, pp. F1D-18-F1D-23. https://doi.org/10.1109/FIE.2008.4720501

[19] C. M. Intisar and Y. Watanobe, "Classification of Online Judge Programmers based on Rule Extraction from Self Organizing Feature Map," in *2018 9th International Conference on Awareness Science and Technology (iCAST)*, 2018, pp. 313–318. https://doi.org/10.1109/ICAwST.2018.8517222

[20] H. Sun, B. Li, and M. Jiao, "YOJ: An online judge system designed for programming courses," in *2014 9th International Conference on Computer Science & Education*, 2014, pp. 812–816. https://doi.org/10.1109/ICCSE.2014.6926575

[21] V. Estivill-Castro, "Concrete Programing for Problem-Solving Skills," *Int. Conf. Educ. New Learn. Technol. (EDULEARN 2010)*, pp. 4189–4197, 2010, [Online]. Available: https://experts.griffith.edu.au/publication/nfb278729c8f2c68cacfecd7cfc5d23e1

[22] C. A. Bagley and C. C. Chou, "Collaboration and the importance for novices in learning java computer programming," *ACM SIGCSE Bull.*, vol. 39, no. 3, pp. 211–215, 2007. https://doi.org/10.1145/1269900.1268846

[23] A. Chorfi, D. Hedjazi, S. Aouag, and D. Boubiche, "Problem-based collaborative learning groupware to improve computer programming skills," *Behav. Inf. Technol.*, vol. 0, no. 0, pp. 1–20, 2020. https://doi.org/10.1080/0144929X.2020.1795263

[24] L. Castillo-Cuesta, C. Ochoa-Cueva, and P. Cabrera-Solano, "Virtual Workspaces for Enhancing Collaborative Work in EFL Learning: A Case Study in Higher Education," *Int. J. Emerg. Technol. Learn.*, vol. 17, no. 2, pp. 4–18, 2022. https://doi.org/10.3991/ijet.v17i02.25937

[25] D. Preston, "Using collaborative learning research to enhance pair programming pedagogy," *ACM SIGITE Newsl.*, vol. 3, no. 1, pp. 16–21, 2006. https://doi.org/10.1145/1113378.1113381

[26] L. Silva, A. J. Mendes, and A. Gomes, "Computer-supported collaborative learning in programming education: A systematic literature review," *IEEE Glob. Eng. Educ. Conf. EDUCON*, vol. 2020-April, no. May, pp. 1086–1095, 2020. https://doi.org/10.1109/EDUCON45650.2020.9125237

[27] A. Radermacher and G. S. Walia, "Investigating the Effective Implementation of Pair Programming : An Empirical Investigation," *SIGCSE'11*, pp. 655–660, 2011. https://doi.org/10.1145/1953163.1953346

[28] G. Braught, L. M. Eby, and T. Wahls, "The effects of pair-programming on individual programming skill," *SIGCSE'08 - Proc. 39th ACM Tech. Symp. Comput. Sci. Educ.*, pp. 200–204, 2008. https://doi.org/10.1145/1352135.1352207

[29] F. J. Rodríguez, K. M. Price, and K. E. Boyer, "Exploring the pair programming process: Characteristics of effective collaboration," *Proc. Conf. Integr. Technol. into Comput. Sci. Educ. ITiCSE*, pp. 507–512, 2017. https://doi.org/10.1145/3017680.3017748

[30] D. Tsompanoudi, M. Satratzemi, and S. Xinogalos, "Exploring the effects of collaboration scripts embedded in a distributed pair programming system," *Annu. Conf. Innov. Technol. Comput. Sci. Educ. ITiCSE*, no. June 2014, pp. 225–230, 2013. https://doi.org/10.1145/2462476.2462500

[31] A. Bhargava, A. Verma, and Satinder, "SOCIAL MOBILITY ANALYTICS CLOUD (SMAC): AN OVERVIEW AND ITS IMPACTS ON SOCIETY," vol. 8354, no. 4, pp. 417–422, 2015.

[32] Z. Li, "Influence of Online Learning Behavior and Video Playing Questions on Students' Learning Effect," *Int. J. Emerg. Technol. Learn.*, vol. 17, no. 2, pp. 223–238, 2022. https://doi.org/10.3991/ijet.v17i02.28535

[33] L.-C. C. Wang, "Student Perceptions of Using Instant Messaging Software to Facilitate Synchronous Online Class Interaction in a Graduate Teacher Education Course," *J. Comput. Teach. Educ.*, vol. 25, no. 1, pp. 15–21, 2008.

[34] F. Ozdamli and N. Cavus, "Basic elements and characteristics of mobile learning," *Procedia - Soc. Behav. Sci.*, vol. 28, no. December, pp. 937–942, 2011. https://doi.org/10.1016/j.sbspro.2011.11.173

[35] A. J. Ko, T. D. LaToza, and M. M. Burnett, "A practical guide to controlled experiments of software engineering tools with human participants," *Empir. Softw. Eng.*, vol. 20, no. 1, pp. 110–141, 2013. https://doi.org/10.1007/s10664-013-9279-3

[36] J. Seanosky *et al.*, "Real-Time Visual Feedback: A Study in Coding Analytics," *Proc. - IEEE 17th Int. Conf. Adv. Learn. Technol. ICALT 2017*, pp. 264–266, 2017. https://doi.org/10.1109/ICALT.2017.38

[37] C. Viegas, G. Alves, and N. Lima, "Formative assessment diversity to foster students engagement," *Proc. 2015 Int. Conf. Interact. Collab. Learn. ICL 2015*, no. October, pp. 929–935, 2015. https://doi.org/10.1109/ICL.2015.7318152

[38] D. Azcona and A. F. Smeaton, "Targeting at-risk students using engagement and effort predictors in an introductory computer programming course," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10474 LNCS, pp. 361–366, 2017. https://doi.org/10.1007/978-3-319-66610-5_27

[39] A. Eckerdal and M. Thuné, "Analysing the Enacted Object of Learning in Lab Assignments in Programming Education," in *2013 Learning and Teaching in Computing and Engineering*, 2013, pp. 208–211. https://doi.org/10.1109/LaTiCE.2013.25

[40] H. Nakayama, K. Ishiwada, Y. Morimoto, S. Nakamura, and Y. Miyadera, "Methods for analysing the editing-processes of source codes in programming exercise for estimating learning situations," *2017 IEEE Conf. e-Learning, e-Management e-Services, IC3e 2017*, pp. 79–84, 2018. https://doi.org/10.1109/IC3e.2017.8409242
[41] F. D. Pereira *et al.*, "Early Dropout Prediction for Programming Courses Supported by Online Judges," in *Artificial Intelligence in Education*, 2019, pp. 66–72. https://doi.org/10.1007/978-3-030-23207-8
[42] M. S. I. M. Zain and S. A. Rahman, "Challenges of Applying Data Mining in Knowledge Management towards Organization," *Int. J. Acad. Res. Bus. Soc. Sci.*, vol. 7, no. 12, pp. 405–412, 2018. https://doi.org/10.6007/IJARBSS/v7-i12/3621
[43] S. Iqbal Malik and J. Coldwell-Neilson, "Impact of a New Teaching and Learning Approach in an Introductory Programming Course," *J. Educ. Comput. Res.*, vol. 55, no. 6, pp. 789–819, 2017. https://doi.org/10.1177/0735633116685852
[44] T. D. Loboda, J. Guerra, R. Hosseini, and P. Brusilovsky, "Mastery Grids: An Open Source Social Educational Progress Visualization," vol. 8719, no. October, 2014. https://doi.org/10.1007/978-3-319-11200-8
[45] R. V. Dorneles, D. P. Jr, and A. G. Adami, "ALGOWEB : A Web-based Environment for Learning Introductory Programming," pp. 4–6, 2010. https://doi.org/10.1109/ICALT.2010.30
[46] S. A. Suriazdin *et al.*, "Technology Attractiveness and Its Impact on MOOC Continuance Intention," *Int. J. Emerg. Technol. Learn.*, vol. 17, no. 04, pp. 239–250, 2022. https://doi.org/10.3991/ijet.v17i04.28853
[47] M. H. Tibi, "Improving Collaborative Skills by Computer Science Students through Structured Discussion Forums," *J. Technol. Educ.*, vol. 10, no. 3–4, pp. 27–41, 2015. https://doi.org/10.4324/9781315704760-17
[48] G. Braught, J. MacCormick, and T. Wahls, "The benefits of pairing by ability," in *SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education*, 2010, pp. 249–253. https://doi.org/10.1145/1734263.1734348
[49] J. Carver, L. Henderson, L. He, J. Hodges, and D. Reese, "Increased retention of early computer science and software engineering students using pair programming," in *Proceedings of the 20th Conference on Software Engineering Education & Training*, 2007, pp. 115–122. https://doi.org/10.1109/CSEET.2007.29
[50] B. Estácio *et al.*, "Evaluating Collaborative Practices in Acquiring Programming Skills: Findings of a Controlled Experiment," in *2015 29th Brazilian Symposium on Software Engineering*, 2015, pp. 150–159. https://doi.org/10.1109/SBES.2015.24

9 Authors

Salihin Rahim is a PhD candidate at School of Computing and Informatics, Universiti Teknologi Brunei (UTB). He was awarded a BSc (Hons) in Computing in 2018 and MSc in Computer and Informatics in 2019 from Universiti Teknologi Brunei. His research interest is in online collaborative educational technology and social media.

Saiful Omar is currently taking up the role of Assistant Vice-Chancellor (Industry & Services) at UTB. He earned his Bachelor's degree in Information Technology from Teesside University in 1998, his Master's degree in Multimedia and Internet Computing in 2004 and PhD in Computer Science in 2014 at Loughborough University. His research and development interest is in Business Process Management, Information

Systems, Virtual Reality, Digital Transformation and Educational Technology. He is currently leading the 5G Innovation Lab to develop 5G use cases on remote video motoring, AR/VR and Smart Campus applications (email: saiful.omar@utb.edu.bn).

Dr Au Thien Wan is a Senior Assistant Professor and the Dean of the Graduate Studies and Research Office, UTB. He earned his BEng Electrical and Electronics Engineering from University of Glasgow, UK, MSc in Data Communications Systems from Brunel University, UK, and PhD from University of Queensland, Australia. His research includes disaster management, sensor networks, edge and fog computing, Software Defined Network (SDN), Internet of Things, Android System, Intelligent Agent System and eLearning experiential learning. His research works have been published in more than 50 papers in academic journals and conferences. He has also completed a few projects such as “Providing an Efficient and Reliable ICT Infrastructure for Smart Grid Data Analytics through Fog Computing”, “Mathematical Wall” and “Development of Software Agent-based Smart Prepaid Wireless Energy Meter”. He has also been invited as a keynote and guest speaker at conferences (email: twan.au@utb.edu.bn).

Irwan Mashadi Mashud is a senior lecturer in School of Computing and Informatics, Universiti Teknologi Brunei. He earned his Bsc (Hons) in Information Technology from University of Teesside, UK, and Master in Software Engineering from Universiti Malaya, Malaysia (email: irwan.mashud@utb.edu.bn).

Article submitted 2022-02-06. Resubmitted 2022-04-13. Final acceptance 2022-04-19. Final version published as submitted by the authors.