Goal-Setting, Active Practice, Self-Monitoring: A Web-Based System to Improve Programming Proficiency

Libor Zachoval, Daire O Broin and Ken Power
South East Technological University (SETU), Carlow, Ireland

libor.zachoval@setu.ie
daire.obroin@setu.ie
ken.power@setu.ie

Abstract: Mastering programming fundamentals poses a significant challenge for students, especially in demanding higher education environments. This pilot study investigates the effectiveness of a web-based system, SoftwareSkills, designed to enhance programming proficiency among second-year bachelor students. SoftwareSkills leverages, active practice, and self-monitoring to support programming practice. Thirty-six students (n=36) participated in a quasi-experimental design, divided into independent exploration and instructional session groups. User interaction data, exercise performance, and behavioural patterns were analysed. The study revealed four distinct learning strategies: reviewing skill masteries, dedicating sufficient time to each question, striving for mastery, and adapting spaced practice. However, some students exhibited rapid responses and minimal practice, suggesting a need for deeper learning strategies. These findings highlight the suitability of self-regulated learning and deliberate practice in programming education. Future work will involve first and second-year programming students and integrate AI-driven personalised learning features to optimise the learning experience. This research contributes insights into effective practices and the potential of technology-aided learning to support student success in programming education.

Keywords: Deliberate practice, Higher education, Programming fundamentals, Self-Regulated learning, Web-Based systems

1. Introduction

Many students who study computing-based courses, particularly software development, encounter problems understanding programming fundamentals (Matthews et al., 2009; Nandigam and Bathula, 2013; Patil et al., 2022). These challenges stem from the multifaceted nature of programming, encompassing core concepts like conditional statements, loops, functions, and variables. Beyond these basics, programming requires strong problem-solving skills and extends to debugging, testing, maintenance, and documentation. Moreover, additional factors such as syntax, code quality, best practices, and software engineering principles further contribute to the overall complexity (Medeiros et al., 2019). Furthermore, the higher education environment can exacerbate these difficulties. Juggling multiple modules each semester forces students to divide their focus, potentially limiting the time available for in-depth learning and skill development. Additionally, individual learning paces necessitate varying levels of time, resources, and support which can challenge lecturers to accommodate effectively (Medeiros et al., 2019).

A strong foundation in these core concepts transcends the need for merely passing exams; it equips individuals with essential building blocks for success in various fields. Programming proficiency empowers individuals to automate tasks, analyse data, and design innovative solutions across diverse industries. In today's increasingly technological world, the ability to think logically, solve problems algorithmically, and translate those solutions into code remains a valuable asset, regardless of career path.

Educators and researchers have explored various technology-aided learning strategies to address these challenges and improve learners' understanding of programming fundamentals. These include Intelligent Tutoring Systems (ITSs), Open Learner Models (OLMs), and Self-Regulated Learning (SRL) frameworks that personalise the learning experience and offer targeted support. This paper presents the findings of a pilot study that explored how learners interacted with a web-based system, SoftwareSkills, which leverages active practice, self-monitoring, and personalised feedback to support programming practice and guide preparation for class tests. The study particularly investigated beneficial strategies learners employed while using the system.

2. Literature Review

2.1 Technology-Aided Learning Strategies for Programming

Educators and researchers have explored various strategies and tools to address the challenges faced by students in programming. One approach focuses on tackling specific learning difficulties through techniques

like visualisation tools and problem-solving methodologies, which make programming concepts more intuitive, particularly for those struggling with syntax (Patil et al., 2022). Intelligent Tutoring Systems (ITSs) offer another avenue for personalised learning, addressing the limitations of one-size-fits-all teaching methods by adapting to individual needs and offering targeted challenges and feedback (Orhun, 1989; Butz et al., 2004).

Open Learner Models (OLMs) empower learners to become self-directed by externalising learner data into comprehensible formats (Thomson and Mitrovic, 2010; Bull, 2020). These visual representations of knowledge mastery and performance promote reflection and metacognition, enabling students to critically analyse their strengths and weaknesses and develop more effective learning strategies (Thomson and Mitrovic, 2010; Suleman et al., 2016; Bull, 2020).

2.2 Self-Regulated Learning (SRL)

SRL empowers learners to take control over their educational journey, a crucial skill in programming (Vieira et al., 2018; Ceron et al., 2021) as success in programming hinges on breaking down complex problems, adhering to deadlines, and critically evaluating code. Web-based platforms that personalise the learning experience and offer targeted support can significantly enhance SRL, especially for students who find traditional methods challenging (Niemi et al., 2003). These platforms can foster self-regulation by allowing students to track their progress, identify areas needing improvement, and adjust their study strategies accordingly (Azevedo et al., 2010). This reinforces the positive correlation between self-assessment and better learning outcomes observed in programming education (Azevedo et al., 2010).

2.3 Panadero's Model of SRL

This research adopts Panadero's model (Panadero, 2017) as a foundation for understanding SRL in programming education. Panadero's model integrates elements from various established SRL models, providing a comprehensive framework that encompasses cognitive, metacognitive, motivational, and emotional aspects of learning across three key phases:

- Forethought Phase: Learners define goals for their programming tasks, engage in strategic planning, and assess their motivation and self-efficacy, which are crucial for initiating and sustaining engagement in learning (Pintrich, 1995).
- Performance Phase: Learners implement their planned strategies, actively monitoring their progress through self-monitoring and self-control, enabling real-time adjustments to their learning approach (Weinstein et al., 2011).
- Self-Reflection Phase: Learners evaluate their performance and the effectiveness of used strategies, reflecting on their learning outcomes compared to initial goals and identifying areas for improvement (Bozpolat, 2016).

2.4 Ericsson's Contribution to SRL

Ericsson's (Ericsson et al., 1993; Nandagopal and Ericsson, 2012; Ericsson and Harwell, 2019) work on deliberate practice has significantly influenced our understanding of how learners develop and apply strategies to achieve expert performance. His contributions emphasise the strategic regulation of learning activities, focusing on three key aspects highly relevant to SRL:

- Strategy Presence: Effective learning involves a repertoire of strategies. Ericsson's model highlights the importance of developing comprehensive study plans, actively monitoring learning processes, and regulating learning environments to enhance self-efficacy (Bransen and Govaerts, 2020). Nandagopal and Ericsson's (2012) study found that successful learners actively utilise a wider range of strategies.
- Strategy Frequency: Regular implementation of these strategies is essential. Frequent use of time management and problem-solving techniques fosters engagement and skill refinement through deliberate practice (Nandagopal and Ericsson, 2012).
- Strategy Consistency: Consistent application of SRL strategies across different contexts is crucial for long-term learning goals. Consistent practice fosters robust metacognitive awareness, enabling learners to make informed decisions about their learning processes (Nandagopal and Ericsson, 2012).

2.5 Learner Strategies in Web-Based Learning

Learner strategies are crucial, offering insights into how students develop approaches for mastering programming skills. For example, learners who actively engage with interactive elements like coding exercises, quizzes, and simulations tend to develop better problem-solving skills and a deeper understanding of

programming concepts (Chi, 2009). Furthermore, learners who consistently practice coding tasks and receive immediate feedback are more likely to correct errors and refine their understanding of complex concepts (Ericsson et al., 1993).

Dedicated web-based systems designed specifically for programming practice offer several benefits to learners. These systems complement traditional learning methods by providing opportunities for deliberate practice, a key component of skill acquisition (Ericsson et al., 1993). Learner interaction with these systems can also foster the development of self-regulation skills, such as self-monitoring and self-assessment (Azevedo et al., 2010). Finally, by analysing learner interaction patterns, educators can gain valuable insights to inform teaching practices and personalise the learning experience for future students.

2.6 Research Questions

The central research questions:

- What specific learner strategies are observed during interaction with the system?
- Can learner interactions with the system be categorised into strategies for mastering programming skills?

3. Methodology

We developed SoftwareSkills, a web-based system which allows users to view their skill masteries for various skills (e.g. functions, variables, loops, and conditional statements) belonging to a higher level skill area (e.g. procedural programming), set focus on specific skills they wish to improve and practise those skills by undergoing a series of exercises (5 questions per practice). The system was available to 36 (n=36) 2nd-year bachelor students enrolled in a software engineering module, capturing their interactions with the system and practice-related performance.

This study employed a quasi-experimental design. The class was divided into two groups, each assigned a specific topic (either code smells or user stories) using SoftwareSkills. Independent exploration commenced on November 15th. Due to observed low engagement, an additional dedicated class session was held on November 29th to provide further instruction and support, enhancing the utilisation of the tool. The course structure included graded weekly quizzes, worth 1% each. On December 6th, the understanding of both groups regarding their respective topics was evaluated through an exam.

4. Findings

The analysis of learner data revealed four distinct strategies that may influence learning outcomes. These strategies were categorised with each phase presented in Panadero's model, see Table 1. Each strategy aligns with key components of the SRL cycle: for instance, viewing skills before, during, or after a session incorporates task analysis, self-monitoring, and self-evaluation. Spending at least 10 seconds per question reflects task analysis, self-control, and self-judgement. Achieving 100% mastery at least once involves goal setting, self-motivation, and self-satisfaction. Lastly, participating in more than five sessions demonstrates strategic planning, self-discipline, and reflective self-evaluation. These alignments suggest a structured approach to self-regulated learning, promoting enhanced academic performance through deliberate practice and reflection. However, a significant limitation of this study is the small sample size (n=36), which limits the generalisability of these findings. Future studies should aim to include a larger and more diverse participant group to validate the results across different contexts.

Table 1: Alignment of Identified Learning Strategies with Phases of Panadero's SRL Model.

Strategy	Forethought Phase	Performance Phase	Self-Reflection Phase
Viewing skills before, during, or after a practice session.	Task Analysis, Goal Setting	Self-Monitoring, Self-Control	Self-Evaluation, Reflection, Self-Judgement
Dedicating enough time to each question.	Task Analysis, Strategic Planning	Self-Control, Self-Monitoring	Self-Evaluation, Self-Judgement
Practising until mastery was achieved (100% on a skill).	Goal Setting, Strategic Planning	Self-Motivation, Persistence	Self-Evaluation, Self-Satisfaction, Self-Judgement

Strategy	Forethought Phase	Performance Phase	Self-Reflection Phase
Using spaced repetition to practice.	Strategic Planning, Goal Setting	Self-Discipline, Self-Monitoring	Self-Evaluation, Reflection, Self-Judgement

4.1 Learner Interaction and Practice Patterns

Examining how learners interacted with the system from the beginning of the study and beyond its conclusion (see Figure 1) revealed insights into their exercise practice and reflection on skill development. This analysis suggests some students independently adopted spaced practice, even though neither the system nor the lecturer prompted them to use it. In Figure 1, we observed that 25% (n=9) revisited their skills approximately 250 hours (~10 days) into the study, 11.1% (n=4) revisited their skills around 100 hours (~4 days) in, and 19.4% (n=7) returned at least once more after the dedicated session.

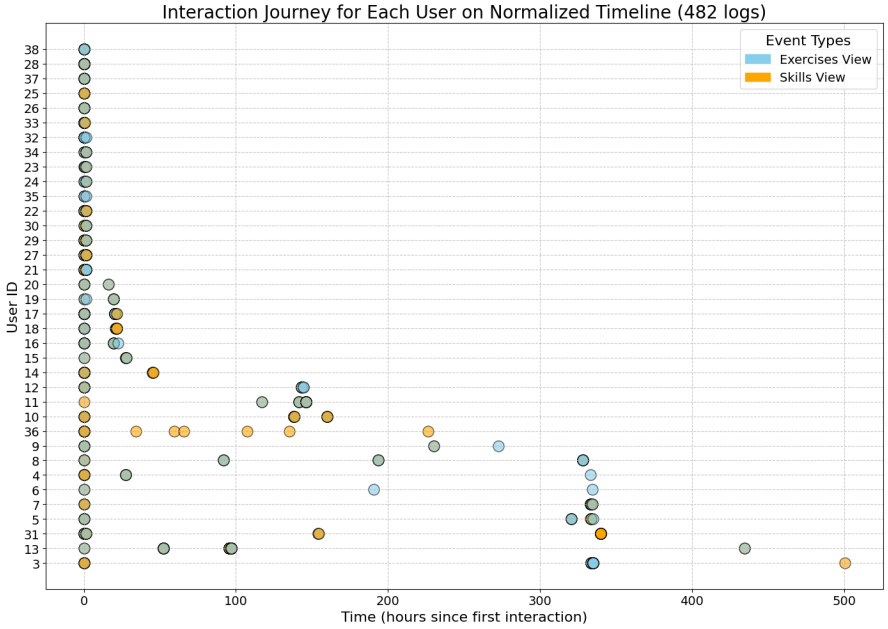

Figure 1: Illustrates varying patterns of spaced repetition, as exemplified by Users 8 and 13, and that certain learners revisited practising specific skills even after the dedicated session

Initial interactions with the system revealed that most learners (88%, n=32) started by exploring the "skill mastery" page, suggesting a focus on understanding their skill gaps or what might be available to them. However, some students (e.g. User IDs 28, 14, and 38) were found alternating between practice sessions and reviewing their skill mastery, before attempting to interact with an exercise. This insight suggests that the learner might have been initially confused about the mechanics.

4.2 Interaction Patterns

We identified four distinct patterns of learner interactions with the system, illustrating varying approaches to viewing skill mastery before and after practice sessions, ordered by popularity (see Figure 2):

- Reviewing skill masteries upon completing a practice session: This was the most common interaction, where learners reviewed their skill mastery after finishing their practice, allowing them to assess their performance immediately.
- Reviewing skill masteries during a sequence of practice sessions: In this pattern, learners took breaks between practice sessions to reflect on their progress, perhaps to determine if further practice is necessary.
- Reviewing skill masteries before initiating a practice session: Some learners preferred to review their skill levels before starting a new practice session, perhaps to guide their focus or set goals.
- Not reviewing skill masteries at any point: The least common pattern involved learners who did not engage in reviewing before, during, or after practice sessions, potentially missing out on valuable insights into their learning progress.

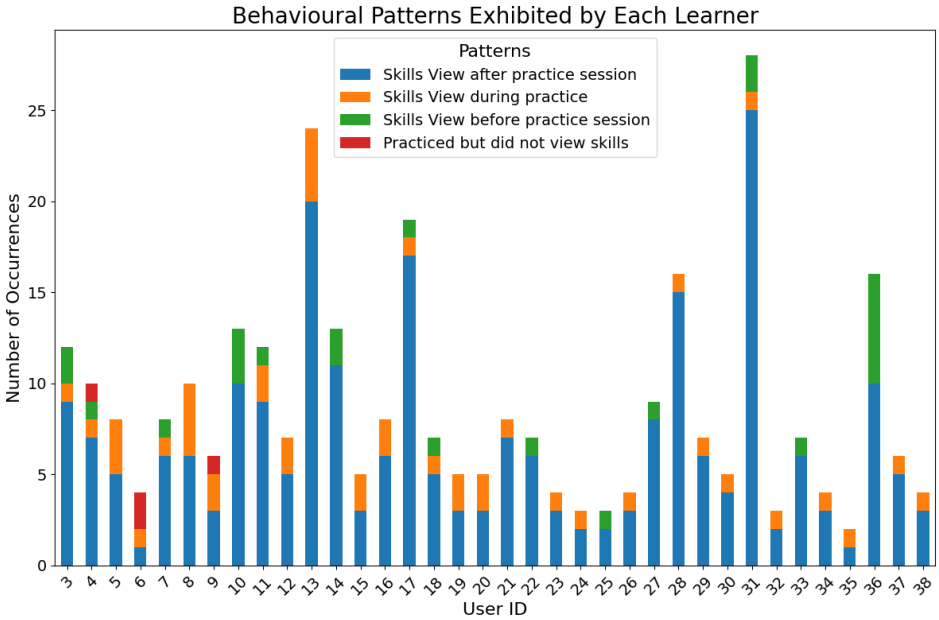

Figure 2: Different interactive patterns identified in each learner

4.3 Mastery Levels

Mastery levels, defined as achieving 100% on either skill (code smell or user stories), varied among learners. As shown in Figure 3, 67% of learners (n=24) achieved mastery at least once, 8% of learners (n=3) reached 50% mastery from a single practice session, and 25% of learners (n=9) did not participate in a single practice session. Of the 24 learners who achieved mastery at least once, 42% (n=10) concluded their practice had reached mastery, whereas 50% (n=5) achieved mastery more than once.

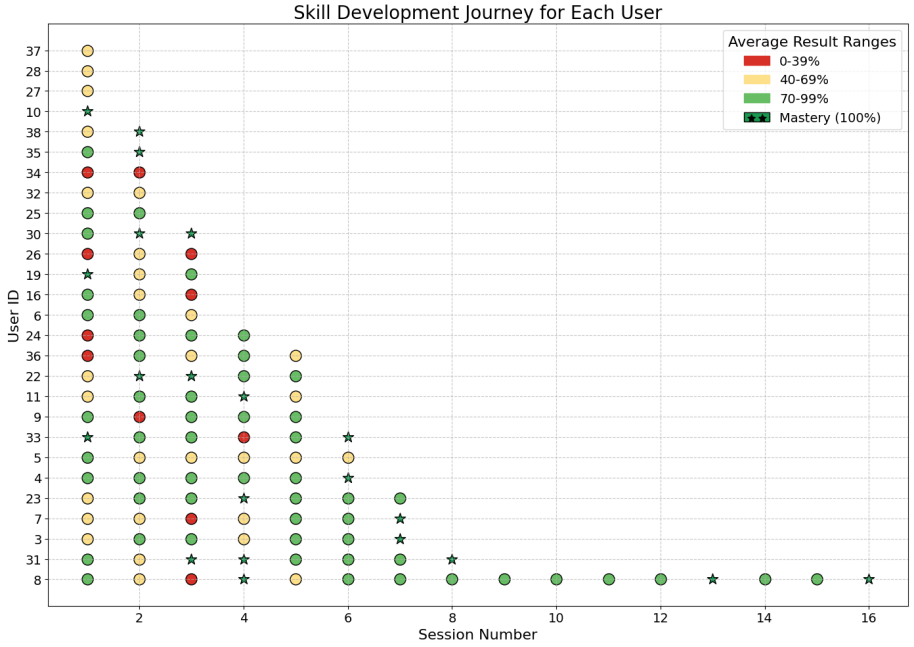

Figure 3: Mastery progression for each user who interacted with the system, a star represents mastery (achieving 100%)

4.4 Response Time Analysis

A deeper analysis of learners' mastery, specifically focusing on the duration spent on each question, revealed a tendency for quick responses (see Figure 4). A significant portion of the questions were answered within 10 seconds. This rapid response rate raises concerns about superficial engagement with the material, potentially

hindering deep learning and comprehensive understanding. Addressing this issue is crucial, as quick responses can undermine the benefits of practice by promoting guessing over genuine problem-solving.

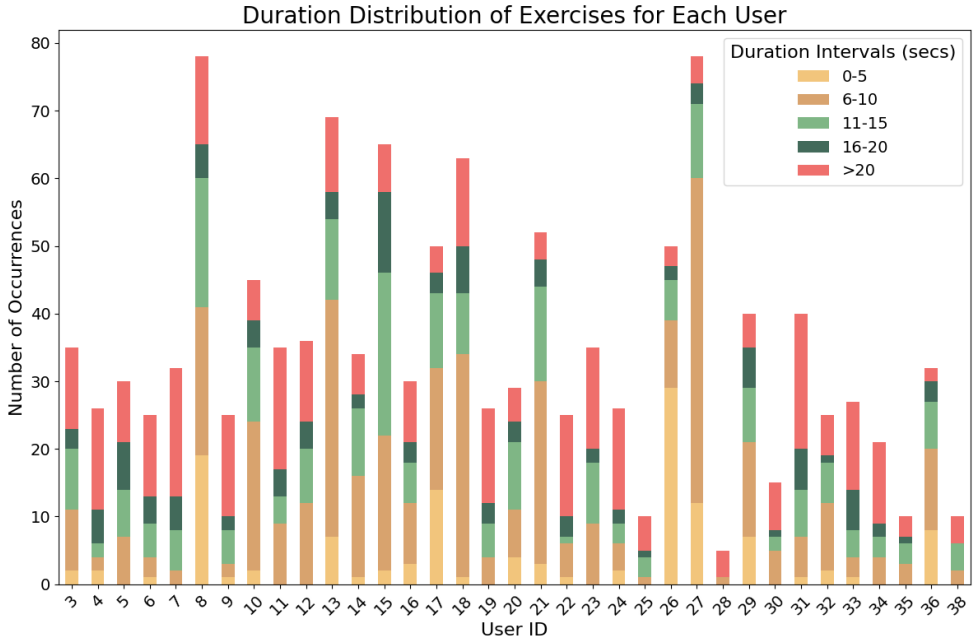

Figure 4: The time taken for each learner to submit their answer to each question encountered

4.5 Session and Attempt Data

Figure 5 provides detailed information on the number of sessions, the number of attempts, and the accuracy of those attempts, including potential guesses (indicated by a response time of less than 10 seconds on the first encounter with a question). The data shows that students who made more attempts did not necessarily guess more often.

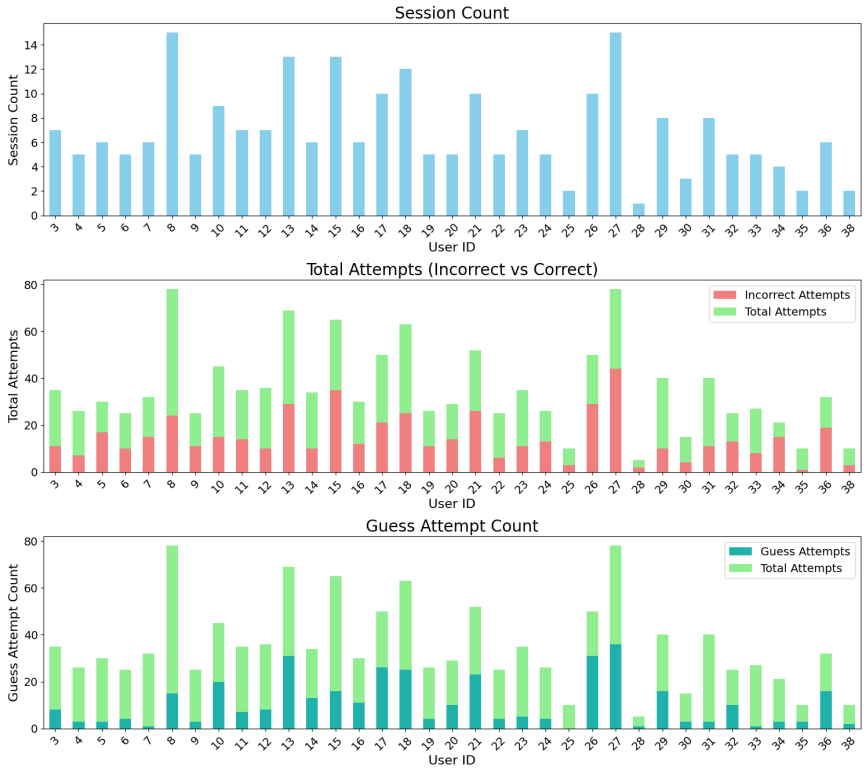

Figure 5: The data includes the number of sessions, the ratio of incorrect attempts to total attempts, and the ratio of correct guesses to total attempts for each learner

4.6 Positive vs. Negative Strategies Ratio

Finally, having explored the different strategies in more detail (e.g., reflecting on skill masteries and taking the time to reach mastery), categorised as positive or negative. The positive-to-negative ratio was estimated at 0.714 vs 0.286 (see Figure 6) using the criteria as defined in Table 2, suggesting that most learners exhibited strategies conducive to learning.

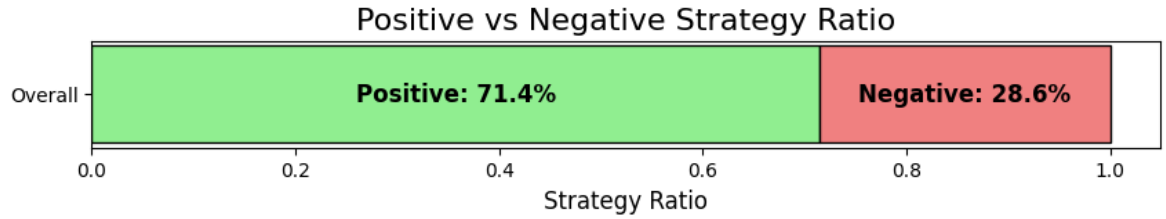

Figure 6: Calculated positive vs negative ratio for learning strategies

Table 2: Criteria for measuring the ratio between positive and negative strategies.

Positive Strategy	Negative Strategy
Viewing skills before, during or after a practice session.	Practising without viewing skill masteries.
Spending at least 10 seconds per question (dependent on the exercise).	Guessing attempts (spending less than 10 seconds on a question).
Achieving 100% mastery at least once.	Never achieving 100% skill mastery.
Participating in more than 5 sessions (ideally spaced days apart from each other).	Participating in less than 5 sessions or closely spaced sessions.

5. Conclusion

This pilot study identified four strategies that may influence learning outcomes in programming education: reviewing skill masteries, dedicating sufficient time to each question, striving for mastery, and adapting spaced practice. These strategies align with the principles of SRL and deliberate practice, highlighting the importance of metacognitive strategies and thoughtful engagement with learning materials. The identification of these strategies provides valuable insights into how learners approach skill mastery in web-based learning environments. Notably, the most common pattern involved reviewing skill mastery after practice sessions, suggesting that immediate self-assessment can deepen learners' understanding of their strengths and weaknesses. However, a subset of learners did not review their skills at any stage, indicating an inability to identify areas needing improvement and adapt their learning strategies. Integrating features that prompt reflection and self-assessment could enhance the learning experience and outcomes within SoftwareSkills.

Moreover, the study revealed that a significant proportion of learners achieved mastery at least once, demonstrating the potential effectiveness of the web-based system in supporting skill acquisition. Some students independently adopted spaced practice without prompts, suggesting that reminding students about spaced practice techniques could further increase their use. However, the study also highlighted a tendency among some learners to respond quickly to questions, often within 10 seconds. This rapid response rate raises concerns about superficial engagement with the material, potentially hindering deep learning and understanding. Addressing this issue is crucial, quick responses can undermine the benefits of practice by promoting guessing over genuine problem-solving. Future iterations of SoftwareSkills could incorporate features that prompt learners to spend more time on each question, such as detailed feedback mechanisms.

In conclusion, while the small sample size and limited duration of the study are notable limitations, the findings provide valuable insights into learner strategies in web-based learning environments. Future research should focus on addressing these limitations by including larger and more diverse participant groups and extending the study period. Additionally, incorporating qualitative data collection methods, such as interviews or focus groups, would provide a richer, more nuanced understanding of learner behaviours and preferences.

6. Limitations

This pilot study provides valuable insight, but some limitations should be considered when interpreting the findings. The study involved a relatively small sample size of 36 second-year bachelor students. This small sample size may not fully represent the diverse range of learners in computing-based courses. Future studies with larger and more diverse populations are necessary to generalise the findings more broadly. The study was conducted over a limited period of time, which may not have allowed enough time to observe long-term learning strategies (e.g. spaced practice) and mastery. Longer-term studies could provide a more comprehensive understanding of how strategies evolve and their impact on learning outcomes.

Despite offering a dedicated class session to enhance tool utilisation, some learners exhibited low engagement with the system. This variation in engagement levels might have influenced the results, potentially underestimating the effectiveness of SoftwareSkills for those who did not fully utilise it. While the analysis primarily focused on quantitative metrics such as time spent on questions and frequency of practice, it did not capture the qualitative aspects of learner experiences and the underlying reasons for certain strategies. Incorporating qualitative research methods, such as interviews, focus groups, or open-ended surveys, would provide a richer, more nuanced understanding of how students perceive and interact with the learning tool.

7. Future Work

We plan to conduct a follow-up study with first and second-year programming students, integrating the system into their curriculum. This follow-up study will address the sample size limitation by including a larger and more diverse cohort of students to validate the results and enhance the generalisability of the findings. The system will become available to students as they cover specific topics in class, with first-year students unlocking access to specific exercises only after they have been covered in lectures, while second-year students will have access to the entire system, having covered the fundamentals in the previous year. This study will span the entire educational year to address the limitation regarding the duration, providing a more comprehensive understanding of how learning strategies evolve over time.

The targeted skills will range from programming basics such as loops to more advanced concepts like classes. Second-year students will take a pre-test before utilising the system to establish a baseline understanding and identify any existing knowledge gaps or misconceptions. This pre-test allows us to tailor the learning experience to each student's individual needs. Following the pre-test, students will be encouraged to use the system to develop their skills towards mastery. Upon achieving mastery in all skill areas, as determined by the system's benchmarks, students will complete a post-test designed to challenge them with complex questions.

The data collected in the follow-up study will include user-system interactions such as viewing skills, focusing on specific skills, engaging in deliberate practice, and accessing resources available within the system. Additionally, the data will record each response, result (correct/incorrect), and duration for each question. This data will be analysed for strategies and their impact on knowledge development. To balance quantitative with qualitative data, all learners will receive a feedback questionnaire at the end of the study to gauge the system's usability, applicability, relevance, and user experience. Incorporating qualitative methods such as these will provide a richer, more nuanced understanding of student behaviours and preferences, potentially enriching the findings with insights that quantitative data alone cannot provide.

To further enhance the learning experience and address the identified need for personalised learning approaches, we plan to develop the following features:

- Progress Trackers: Implement trackers that display days of practice in a calendar format and highlight ongoing streaks. This visual representation can help students maintain consistent practice and monitor their progress over time.
- Interactive Skill Map: Display an interactive map of all skills to highlight the overall development of knowledge. This map can help students visualise their progress and understand the interconnectedness of different programming concepts.
- Prompts and Notifications: Regularly prompt learners with reminders to practice, encouraging more thoughtful engagement with exercises and adoption of more positive strategies like spaced practice.

The varied levels of engagement and mastery observed highlight the need for personalised learning approaches. To address this, we are considering the integration of Artificial Intelligence (AI) through the following features:

- Learning Path Guidance: This feature will recommend the next steps based on each student's learning progress, ensuring they focus on areas requiring improvement and navigate through the material effectively.
- AI Mentor: An AI mentor will provide students with in-depth personalised feedback on their data, performance, strengths, and weaknesses. This feedback will guide students in optimising their learning strategies and addressing specific challenges.

References

Azevedo, R., Moos, D.C., Johnson, A.M., Chauncey, A.D., 2010. Measuring Cognitive and Metacognitive Regulatory Processes During Hypermedia Learning: Issues and Challenges. Educ. Psychol. 45, 210–223. https://doi.org/10.1080/00461520.2010.515934

Bozpolat, E., 2016. Investigation of the Self-Regulated Learning Strategies of Students from the Faculty of Education Using Ordinal Logistic Regression Analysis. Educ. Sci. Theory Pract. 16, 301–318.

Bransen, D., Govaerts, M.J.B., 2020. How to conceptualise self-regulated learning: Implications for measurement. Med. Educ. 54, 684–686. https://doi.org/10.1111/medu.14183

Bull, S., 2020. There are Open Learner Models About! IEEE Trans. Learn. Technol. 13, 425–448. https://doi.org/10.1109/TLT.2020.2978473

Butz, C.J., Hua, S., Maguire, R.B., 2004. A Web-Based Intelligent Tutoring System for Computer Programming, in: IEEE/WIC/ACM International Conference on Web Intelligence (WI'04). Presented at the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), IEEE, Beijing, China, pp. 159–165. https://doi.org/10.1109/WI.2004.10104

Ceron, J., Baldiris, S., Quintero, J., Garcia, R.R., Saldarriaga, G.L.V., Graf, S., Fuente Valentin, L.D.L., 2021. Self-Regulated Learning in Massive Online Open Courses: A State-of-the-Art Review. IEEE Access 9, 511–528. https://doi.org/10.1109/ACCESS.2020.3045913

Chi, M.T.H., 2009. Active-Constructive-Interactive: A Conceptual Framework for Differentiating Learning Activities. Top. Cogn. Sci. 1, 73–105. https://doi.org/10.1111/j.1756-8765.2008.01005.x

Ericsson, K.A., Harwell, K.W., 2019. Deliberate Practice and Proposed Limits on the Effects of Practice on the Acquisition of Expert Performance: Why the Original Definition Matters and Recommendations for Future Research. Front. Psychol. 10.

Ericsson, K.A., Krampe, R.T., Tesch-Römer, C., 1993. The role of deliberate practice in the acquisition of expert performance. Psychol. Rev. 100, 363–406. https://doi.org/10.1037/0033-295X.100.3.363

Matthews, R., Hin, H.S., Choo, K.A., 2009. Multimedia learning object to build cognitive understanding in learning introductory programming, in: Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia. Presented at the MoMM '09: 7th International Conference on Mobile Computing and Multimedia, ACM, Kuala Lumpur Malaysia, pp. 396–400. https://doi.org/10.1145/1821748.1821824

Medeiros, R.P., Ramalho, G.L., Falcão, T.P., 2019. A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. IEEE Trans. Educ. 62, 77–90. https://doi.org/10.1109/TE.2018.2864133

Nandagopal, K., Ericsson, K.A., 2012. An expert performance approach to the study of individual differences in self-regulated learning activities in upper-level college students. Learn. Individ. Differ., Adult Learning 22, 597–609. https://doi.org/10.1016/j.lindif.2011.11.018

Nandigam, D., Bathula, H., 2013. Competing Dichotomies in Teaching Computer Programming to Beginner-Students. Am. J. Educ. Res. 1, 307–312. https://doi.org/10.12691/education-1-8-7

Niemi, H., Nevgi, A., Virtanen, P., 2003. Towards self-regulation in web-based learning. J. Educ. Media 28, 49–71. https://doi.org/10.1080/1358165032000156437

Orhun, E., 1989. Knowledge Representation in Intelligent Tutoring Systems for Computer Programming. Am. J. Math. Manag. Sci. 9, 243–259. https://doi.org/10.1080/01966324.1989.10737265

Panadero, E., 2017. A Review of Self-regulated Learning: Six Models and Four Directions for Research. Front. Psychol. 8. https://doi.org/10.3389/fpsyg.2017.00422

Patil, Prof.A., Mane, Prof.D., Shinde, Prof.N., 2022. Incorporating Visualization Tools and Active Learning Approach for Programming Courses. J. Eng. Educ. Transform. 35, 92–102. https://doi.org/10.16920/jeet/2022/v35is1/22014

Pintrich, P.R., 1995. Understanding self-regulated learning. New Dir. Teach. Learn. 1995, 3–12. https://doi.org/10.1002/tl.37219956304

Suleman, R.M., Mizoguchi, R., Ikeda, M., 2016. A New Perspective of Negotiation-Based Dialog to Enhance Metacognitive Skills in the Context of Open Learner Models. Int. J. Artif. Intell. Educ. 26, 1069–1115. https://doi.org/10.1007/s40593-016-0118-8

Thomson, D., Mitrovic, A., 2010. Preliminary evaluation of a negotiable student model in a constraint-based its. Res. Pract. Technol. Enhanc. Learn. 05, 19–33. https://doi.org/10.1142/S1793206810000797

Vieira, C., Parsons, P., Byrd, V., 2018. Visual learning analytics of educational data: A systematic literature review and research agenda. Comput. Educ. 122, 119–135. https://doi.org/10.1016/j.compedu.2018.03.018

Weinstein, C.E., Acee, T.W., Jung, J., 2011. Self-regulation and learning strategies. New Dir. Teach. Learn. 2011, 45–53. https://doi.org/10.1002/tl.443