ASCILITE 2023

People, Partnerships and Pedagogies

IDE vs. pen-and-paper showdown - Performance and perceptions of testing first-year IT students' programming skills

David Harris, G. Stewart Von Itzstein, Douglas Kelly and Elizabeth Smith
University of South Australia

This study analysed first-year IT students' performance and perceptions of using Integrated Development Environment (IDE) versus pen and paper (P&P) based programming testing methods. 68 participants completed two programming tests using both testing methods, and the results were evaluated on complexity, quality and other criteria. No significant difference was observed in the average scores between the two test methods. However, IDE testing showed fewer syntax errors and lower cyclomatic complexity, indicating the potential advantages of IDE's error-checking. IDE tests exhibited higher variability in student performance, suggesting varying comfort levels with the test environment. Additionally, a weaker correlation existed between actual and perceived performance in IDE tests, highlighting challenges in self-evaluation within a digital framework. Most students preferred IDE-based testing, citing efficiency, user-friendliness, and real-time feedback. Nonetheless, a minority recognised the value of P&P tests for demonstrating deep syntax understanding. This research enhances our understanding of programming pedagogy and assessment.

Keywords: Programming, Assessment, paper vs. computer

Introduction and background

Teaching programming to first-year students in an Information Technology (IT) degree program can be a challenging task due to several factors. One challenge is the requirement for students to perform higher-level thinking early in the learning process as programming requires an understanding of rigid syntax and commands (Kelleher & Pausch, 2005). Learning programming also necessitates understanding diverse areas like logic, mathematical thinking, algorithms, data flow, abstraction, indirection, etc. which can inundate the learner, leading to heightened stress and frustration, resulting in increased cognitive load (Stachel et al., 2013). This can be particularly difficult for students who are new to programming and may not have prior experience with problem-solving, abstraction, and critical thinking in this domain (Medeiros et al., 2018). Consequently, there is a need for higher-order understanding before lower-order understanding is fully established.

Given the challenges students encounter while learning programming, it is anticipated there will be a rise in the adoption and dependence on generative AI tools such as ChatGPT and Copilot as they become more widely accessible. Consequently, the assessment of authentic individual comprehension of the theoretical concepts behind programming becomes even more critical. The use of computer-based tests, using Integrated Development Environments (IDEs), is a common method for assessing student programming foundation knowledge and it is the students preferred testing method over P&P programming tests (Barros, 2018). P&P programming tests are still used but are perceived as an artificial scenario that may not fully assess students' ability to develop program code (Bennedsen & Caspersen, 2007). Recent research suggests that there is no significant difference in student test performance between using an IDE or a P&P exam (Corley et al., 2020; Öqvist & Nouri, 2018). Consequently, P&P-based tests still offer a suitable measure of students' programming ability and provide various advantages, including evaluating their capacity to reason about code without relying on IDEs, assessing theoretical understanding, reducing susceptibility to plagiarism, and being easier to invigilate compared to IDE tests. For these reasons, P&P programming tests continue to be used in the first year of the IT programs at the University of South Australia (UniSA). While previous research has compared student performance in IDE and P&P programming tests (Corley et al., 2020; Öqvist & Nouri, 2018), as well as examining students preferred testing method (Barros, 2018), the authors of this study aim to further explore students' perceptions of performance between these two testing methods and their relationship with IDEs and the support they provide in comparison to P&P-based testing methods. This study aims to address the following research questions: **RQ1.**How do students perceive their performance in IDE-based programming tests compared to paper-based testing methods? **RQ2.**What is the nature of the relationship between students and IDEs in terms of their perceived support and effectiveness, in contrast to P&P-based testing methods?

Methodology

This study explores how first-year IT students perceive and perform in IDE-based vs P&P programming tests at UniSA. It also explores student views on the effectiveness of these testing methods and their relationship with IDEs in comparison to traditional P&P-based tests. The study was conducted over two semesters, involving 68 student participants. Students were asked to complete two programming tests of comparable complexity. One of these tests was administered via an IDE (IDLE) while the other test used a P&P method. The order of the tests was randomised to minimise any possible order effects, and the P&P and IDE questions were swapped to avoid any question bias dependent on the testing mode. All students from a first-year programming class were invited to partake in the study during the last week of the semester. Tests were performed under exam-like conditions and students were given a maximum of 30 minutes per test. The evaluation of these tests was performed by two independent markers, using a detailed structured rubric to ensure the consistency and impartiality of the scoring process. The rubric allocated marks based on standard marking criteria such as syntax errors, infinite loop occurrences, Big O complexity, cyclomatic complexity, unnecessary variable usage, coding style and layout, and encapsulation. Upon completion of both tests, students were asked to participate in a survey. The survey incorporated Likert scale questions to assess the students' perceived performance on each test, as well as their testing preferences. Additionally, students were prompted to provide open-ended text responses to enable a more nuanced exploration of their experiences and perceptions. These responses were subsequently analysed using thematic analysis to identify patterns and key insights. This paper presents the preliminary findings from this study.

Results and Discussion

Student Test Performance and Perceived Performance

This study compared the performance of students (n=68) sitting similar IDE and P&P-based programming tests. The average score on the IDE test was 11.32 (out of 15), which is slightly lower than the average P&P test score of 11.91 (out of 15). However, a paired two-tail t-test found no statistically significant difference between these means ($t=1.15$, p-value = 0.253), suggesting the observed difference could be due to chance. There is also a modest positive correlation between the scores from the P&P test and the IDE test with a correlation coefficient of 0.242. This suggests that students who performed well on the P&P test generally also performed reasonably well on the IDE test and vice versa. This aligns with the work of Öqvist and Nouri (2018) and Corley et al. (2020) who also found there was no significant difference in students' performance between IDE and P&P tests.

In examining the comparative syntax errors and cyclomatic complexity between IDE vs P&P- test performances of students, this study found the average number of syntax errors was significantly lower when using an IDE (mean = 0.31) compared to P&P (mean = 1.09), as indicated by the t-test ($t(67) = 5.82$, $p < 0.001$). This suggests that students made fewer mistakes when programming in an IDE, likely due to its built-in syntax checking. These results also align with the findings of Corley et al. (2020) who observed students demonstrated fewer syntax errors when using the IDE testing method. Cyclomatic complexity measures the number of linearly independent paths through a program's source code. Ideally, the value should be as low as possible to solve the problem, with a score of one being a single path from beginning to end. In the present study, mean scores for cyclomatic complexity were slightly lower in the IDE testing method (mean = 5.85) than in the P&P testing method (mean = 6.41), $t(67) = 1.82$, $p = 0.073$. This could indicate that the use of an IDE leads to simpler, more streamlined code. This evidence supports the view that IDEs, with their syntax-checking capabilities, offer substantial advantages for student programmers over P&P methods.

In terms of variability in scores between IDE and P&P tests, there was higher variability in scores for the IDE test (standard deviation = 4.02) than the P&P test (standard deviation = 2.56), implying a wider range of student performance on the IDE test. Additionally, score distributions for both tests were negatively skewed, indicating more students scored below the mean. This skew was more pronounced in the IDE test (-1.53 vs. -1.33). The higher variability in scores for the IDE test may indicate that some students are more comfortable or familiar with this testing environment, leading to higher scores, while others may be less comfortable or familiar, leading to lower scores. Thus, the IDE might enhance performance for some students while potentially hindering others. The reason the IDE may hinder some student's performance is due to the possible higher cognitive load associated with IDE overheads of the interface, debugging, and navigating files as suggested by Stachel et al. (2013).

This study also found a moderate positive correlation ($r = 0.412$) between students' actual scores and their perceived performance in traditional P&P tests, indicating that those who performed better also rated their

performance higher. This points to a level of self-assessment accuracy in traditional test settings. Conversely, the correlation between actual scores and perceived performance on the IDE tests was weaker ($r = 0.228$). This suggests that students with higher IDE scores only somewhat rated their performance better, hinting at challenges in self-evaluation within a digital testing framework. This is surprising, considering that one might expect students to be more confident in assessing their performance when they can use the built-in error-checking capabilities of the IDE. However as mentioned above, the IDE test scores had a higher variability, also suggesting some students may struggle with the added complexities of the IDE environment which could also explain why there is a weaker correlation between actual and perceived performance in the IDE setting.

Quantitative Survey Responses

Following completion of the IDE and P&P-based tasks, all students were surveyed on their preferred testing method. Student responses to the survey showed there was a significant preference for the IDE-based testing method, with 92.5% (62 out of 67) of participants expressing this preference. Only 7.5% (5 out of 67) favoured P&P-based testing. These results align with the findings of other studies including Barros (2018). In the survey, students were also asked to evaluate the impact of syntax highlighting on learning to program, a key feature of IDEs. The results show a majority of the participants found syntax highlighting beneficial with 76.5% (52 out of 68) of students reporting that syntax highlighting helped 'a lot' or 'a little', in contrast to 17.6% (12 out of 68) of participants suggesting syntax highlighting offered 'no help at all' or 'very little' help and 5.9% (4 out of 68) were 'unsure' if the syntax highlighting provided any benefit. Consistent with these findings, a significant number of students 75% (51 out of 68) indicated that the IDE test provided more time to consider and develop their responses, compared to 23.5% (16 out of 68) who thought the P&P-based test provided greater opportunity to develop solutions. Moreover, most students also indicated they could develop answers more quickly in the IDE test, with 72.1% (49 out of 68) compared to 27.9% (19 out of 68) indicating a solution was derived more quickly during the P&P-based test. Students were also asked which testing method better reflected their problem-solving abilities and programming skills. The results showed that the majority thought the IDE test better reflected both their problem-solving abilities (66.2%, 45 out of 68) and programming skills (93.8%, 61 out of 65).

Finally, the self-evaluation of performance on both tests was examined. Notably, there were no 'very poor' responses for either test approach. For the P&P-based test, most students felt their performance was 'okay' (36.8%) or 'good' (33.8%), with 'excellent' being the least selected option (13.2%). In contrast, for the IDE test, the majority rated their performance as 'excellent' (39.7%) or 'good' (29.4%), with 'okay' (19.1%) and 'poor' (11.8%) being less common. These results suggest that students generally prefer IDE-based testing and find the syntax highlighting useful, but also that this approach allows for more thoughtful consideration and quicker answers. As a result, participants feel that using an IDE better reflects their problem-solving abilities and programming skills.

Qualitative Survey Responses

To further understand the reasons behind these preferences students were also asked to respond to two text-based questions. Firstly, students were asked to discuss why they selected their preferred testing method (n=63) and secondly, regarding their least preferred test approach, what impacted their ability to answer the question (n=55). For the first question, on why students selected their preferred testing method, a thematic analysis was performed on the 59 text responses received from 92.5% (62 out of 67) of participants who selected the IDE as their preferred testing method. Four main themes were identified and are discussed below along with the percentage of responses that mentioned this theme. **IDE Efficiency and User-friendliness** (100% of responses)**:** All students expressed appreciation for the efficiency and user-friendliness of IDEs. For instance, one student pointed out *"The computer-based test was helpful as it allowed me to easily navigate through and write the code faster without thinking too much about the presentation and indentation."* This quote reveals how an IDE's features can simplify the coding process and allow students to focus more on coding logic rather than formatting or presentation. **Physical and Cognitive Comfort** (69.5% of responses)**:** Physical and cognitive comfort was a recurring theme in the responses. A student stated, *"I type faster than I can write. It's harder to undo mistakes on paper. My wrist hurts. Computers save me from my handwriting."* This response is indicative of how physical comfort can be a significant factor when it comes to choosing between handwriting or typing code. Cognitive comfort also plays a significant role, as illustrated by another student's comment: *"The IDE gives me real time feedback. It is difficult to remember all syntax so having IDE to help with that negates this potential waste of time."*. **Relevance and Applicability** (42.4% of responses)**:** The practical relevance and applicability of using an IDE was also a significant factor for the students. One student stated, *"I think the computer based test best tests a student's ability to program and solve problems in the real world."* This student,

along with many others, recognised the value of being proficient in an IDE given its widespread use in professional settings. **P&P Evaluation** (23.7% of responses): Even though the IDE was the preferred test method, some students found value in P&P tests. As one student noted, *"... both approaches I like as I feel like paper based shows off someone's actual knowledge and syntax without needing any help from the program. It is also slower to write the answer so it gives you time to think and consider the code more."* This quote highlights the perceived benefits of paper-based evaluations, including a slower pace, which allows for more thoughtful consideration of the code, and the opportunity to demonstrate an understanding of syntax without any assistance. These quotes illustrate the primary reasons students prefer IDE-based testing while still acknowledging the potential benefits of P&P evaluations. Those students who preferred the P&P tests (n=4) also provided text responses on why they chose this as their preferred test method. For example, one of the key advantages of P&P-based testing articulated by students is the opportunity to note down and organise their thoughts more efficiently. As one student notes, *"For the paper questions, I was able to break down the questions and write down the key information I have needed to answer question, writing on paper helped me write down ideas on how to answer."* This aspect of handwriting enables an easy visual representation of thoughts, aiding in problem-solving. Another aspect reflected in the responses, is the alleviation of anxiety over minor syntax errors, allowing students to concentrate more on the core logic. One student mentioned, "*Less stressed about minor syntax errors and focused on the problem-solving aspect."* This brings into focus how P&P-based testing can foster a focus on problem-solving, which is the crux of programming.

For the second text-based survey question, on how students least preferred test approach impacted their ability to answer the programming question, a thematic analysis was performed on the 52 text responses received from the participants who selected the IDE as their preferred testing method. Four main themes were identified and are discussed below along with the percentage of responses that mentioned this theme. **Lack of Interactive Coding Features** (59.6% of responses): This theme appeared frequently in the responses, with students expressing frustration about not being able to test and debug their codes in real time. As an example, one student straightforwardly stated, *"Testing code - inability to do this with paper-based."* Similarly, another student expressed, *"Not being able to see if it worked was an impact because I wasn't sure if I completed it correctly."* The lack of real-time testing significantly impacts the students' confidence in their solutions and forces them to work more on assumptions. This represents a departure from the interactive testing environment they are typically accustomed to when using IDEs. **Usability Constraints** (51.9% of responses): Writing codes on paper inherently carries usability constraints such as the inability to modify written text, which proved to be a major source of frustration for students. One student articulated this challenge as, *"Not being able to delete a line of code that I feel I stuffed up, I had to cross it out which sort of broke my focus."* This comment demonstrates how the physical limitations of P&P testing could disrupt students' thought processes and make the process of correcting errors distracting. **Recall and Understanding** (48.1% of responses): The P&P-based approach seemed to place an additional burden on students, as they had to rely more heavily on their memory and understanding of programming concepts without the help of IDEs. One student commented, *"In paper-based, it's more on remembering what you learned from your lessons and applying it to the test questions."* This remark clarifies the need for a deeper understanding of programming concepts and a stronger recall of syntax and functions with P&P-based testing. **Visualisation and Comprehension Challenges** (44.2% of responses): Without the colour-coding and auto-indentation features of IDEs, students found it challenging to visualise and comprehend their codes. One student shared, *"No coloured syntax words like the computer - difficult to remember where indents are."* This remark emphasises the importance of IDE features in enhancing code readability and comprehension, which are vital for effective coding. In addition, another student confessed, *"I think because I am slower to type the code compared to writing it, it seems as though I struggle to visualise my ideas before they disappear."* This underlines the struggles students face in visualising their coding logic and thoughts in the absence of visual aids like colour-coding and auto-indentation. These responses demonstrate how the shift from IDEs to a P&P-based testing approach seems to create significant challenges for students. These challenges underscore the importance of IDE features not only for enhancing coding efficiency but also for supporting students' understanding and problem-solving processes in programming.

For students who preferred the P&P-based testing method, one student commented on their ability to facilitate and capture the thought processes of students. A student revealed, *"If the paper testing was used, the students could write down their thinking if they don't know what exact code they need to use but possibly get marks to show they know what they need to do to get the answer."* This suggests that P&P tests allow for partial credit for problem-solving skills and understanding, even when the exact code escapes the student's memory. Moreover, P&P-based tests seem to alleviate technical issues that can cause undue stress and wasted time during computer tests. One respondent illustrated this point with the quote, *"Syntax errors, making new files ensuring the technology was working just added more time required to get everything working"*. This statement emphasises how technological problems can distract from the main task of demonstrating coding understanding and

problem-solving skills. However, the pitfalls of P&P-based testing are not completely overlooked by these students. They still acknowledge issues such as syntax errors and the challenge of remembering language-specific details as significant drawbacks. While there are clear advantages to using IDEs for testing students' coding abilities, there are also several advantages to continue using P&P-based tests despite the difficulties and drawbacks they pose. For example, **examination integrity** as P&P-based tests minimise the potential for cheating or using unauthorised resources during the exam, **assessment of fundamental understanding** as P&P tests require students to recall & apply concepts without the aid provided in IDEs, and, organising P&P tests can be **easier logistically**, especially for large classes where access to computer pools is an issue

Conclusion

The findings from this study provide insights into students' performance, perception, and preferences when it comes to IDE and P&P-based programming tests. No significant difference was observed in the average scores between the two methods, yet fewer syntax errors and more streamlined code were noted in the IDE testing environment. However, a higher variability in IDE test scores and a weaker correlation between actual and perceived performance suggests some students may grapple with the complexities of the IDE environment, such as understanding the interface, debugging, and navigating files. In addition, the results show most of the participants found IDE syntax highlighting beneficial in terms of reducing errors and better reflecting their problem-solving abilities and programming skills. The research questions were addressed in this study with RQ1 showing that students think they know more and do better with the IDE even if that is not the case in the outcomes of the test. RQ2 showed that students prefer the IDE testing mode even though generally there were no real differences in the outcomes across the testing modes. This study underlines the importance of understanding students' interaction with testing methods and the role of self-assessment in these settings, providing insights for educators and curriculum developers in the field of computer science and programming.

References

Barros, J. P. (2018). Students' perceptions of paper-based vs. computer-based testing in an introductory programming course. CSEDU 2018-Proceedings of the 10th International Conference on Computer Supported Education. Vol. 2. SciTePress, 303–308. https://doi:10.5220/0006794203030308

Bennedsen, J., & Caspersen, M. E. (2007). Assessing process and product: a practical lab exam for an introductory programming course. Innovation in Teaching and Learning in Information and Computer Sciences, 6(4), 183-202. https://doi.org/10.1109/FIE.2006.322434

Corley, J., Stanescu, A., Baumstark, L., & Orsega, M. C. (2020). Paper or ide? the impact of exam format on student performance in a cs1 course. Proceedings of the 51st ACM technical symposium on computer science education. 706–712. https://doi.org/10.1145/3328778.3366857

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys (CSUR), 37(2), 83-137. https://doi.org/10.1145/1089733.1089734

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. IEEE Transactions on Education, 62(2), 77-90. https://doi.org/10.1109/TE.2018.2864133

Öqvist, M., & Nouri, J. (2018). Coding by hand or on the computer? Evaluating the effect of assessment mode on performance of students learning programming. Journal of Computers in Education, 5, 199-219. https://doi.org/10.1007/s40692-018-0103-3

Stachel, J., Marghitu, D., Brahim, T. B., Sims, R., Reynolds, L., & Czelusniak, V. (2013). Managing cognitive load in introductory programming courses: A cognitive aware scaffolding tool. Journal of Integrated Design and Process Science, 17(1), 37-54. https://doi.org/10.3233/jid-2013-0004

Harris, D., Von Itzstein, G. S., Kelly, D. & Smith, E. (2023). IDE vs. pen-and-paper showdown - Performance and perceptions of testing first-year IT students' programming skills. In T. Cochrane, V. Narayan, C. Brown, K. MacCallum, E. Bone, C. Deneen, R. Vanderburg, & B. Hurren (Eds.), *People, partnerships and pedagogies*. Proceedings ASCILITE 2023. Christchurch (pp. 430 - 434). https://doi.org/10.14742/apubs.2023.541