A Systematic Review of Formative Assessment to Support Students Learning Computer Programming

Jagadeeswaran Thangaraj
School of Computing, Dublin City University, Ireland

Monica Ward
School of Computing, Dublin City University, Ireland

Fiona O'Riordan
Teaching Enhancement Unit, Dublin City University, Ireland

Abstract

Formative assessment aims to increase student understanding, instructor instruction, and learning by providing feedback on students' progress. The goal of this systematic review is to discover trends on formative assessment techniques used to support computer programming learners by synthesizing literature published between 2013 and 2023. 17 articles that were peer-reviewed and published in journals were examined from the initial search of 197 studies. According to the findings, all the studies were conducted at the higher education level and only a small number at the secondary school level. Overall, most studies found that motivation, scaffolding, and engagement were the three main goals of feedback, with less research finding that metacognitive goals were the intended outcomes. The two techniques for facilitating formative feedback that were used most frequently were compiler or testing based error messages and customised error messages. The importance of formative feedback is highlighted in the reviewed articles, supporting the contention that assessments used in programming courses should place a heavy emphasis on motivating students to increase their level of proficiency. This study also suggests a formative assessment that employs an adaptive strategy to evaluate the ability level of the novice students and motivate them to learn programming to acquire the necessary knowledge.

2012 ACM Subject Classification Applied computing → Education; Social and professional topics → Computing education; Social and professional topics → Student assessment

Keywords and phrases Automatic assessment, Computer programming, Formative assessment, Higher education, Novice programmer, Systematic review

Digital Object Identifier 10.4230/OASIcs.ICPEC.2023.7

1 Introduction

Any course in a higher education institution worldwide that is concerned with software development requires programming modules. By introducing syntax and semantics, these modules aim to impart fundamental knowledge of programming languages [58]. Novice programmers are those taking their first computer programming courses or those with no prior programming experience. Independent components of programming will increase the difficulties of novices [42, 32]. Novice programmers are unable to interpret program code and have a lack of understanding of programming principles [27]. Although the computer science courses are in high demand, introductory programming modules frequently have dropout and failure rates as high as 50% [36, 34]. These modules play an important role to make them comfortable in continuing their education in computing [46]. Their interest in programming will rise once pedagogical methods motivate their confidence, and dropout rates will reduce [36].

An essential component of education that promotes learning is assessment [48]. Additionally crucial for assisting and motivating their programming abilities are assessment and feedback [57]. Formative assessment aims to increase student understanding, instructor instruction, and learning by providing feedback on students' progress[12]. Formative programming assessment system evaluates student program submissions and provides timely feedback [42]. The impact of feedback on learning and assessment is significant [55]. Formative feedback is information given to a student with the goal of changing their way of thinking or acting to enhance learning [50]. Formative assessment is one of the approaches for effective programming learning [53]. To ensure that students receive the correct results, the assessment and feedback systems must examine the programs' small aspects and point out any areas where mistakes were made by the students. Beyond assessment, it increases novice programmers' self-confidence and metacognitive awareness [41, 49, 29]. Additionally, checking a participant's knowledge depending on their prior attempts during the assessment process is referred to as adaptive assessment [39]. It varies from typical assessment in that each participant receives a separate set of questions instead of everyone receiving the same set of questions [56]. This enables everyone to assess the knowledge at their own pace and helps in assessing each element of the topic. While some systematic reviews have been undertaken on automated assessments in computer programming, this study is interested in knowing whether formative assessment is used to scaffold or encourage novice programmers as feedback is essential to learning programming. It also examines any assessment system that makes use of an adaptive strategy.

2 Related Works

Recent systematic literature reviews on assessment systems for programming courses tend to concentrate on how useful they are for automatic assessment techniques [24, 40, 20] or on the kind of feedback generated by assessment tools for evaluating programming languages or programming paradigms [25, 9, 28]. Further, other research reviews examined the assessment process, finding that it was primarily concentrated on advanced courses [35] or other computing subjects [16]. Studies that use automatic assessment programming as a pedagogical strategy typically focus solely on teaching and learning and do not examine findings on inspiring or motivating novices. In summary, there are no studies that concentrate on the formative assessment of introductory programming courses and no research reports that motivate novice programmers. Since motivation, scaffolding, and metacognitive support are important for learning programming modules, this research therefore examines formative assessment methods in these areas.

Research Questions

This review of formative assessments for introductory programming is guided by the following research questions:

- RQ1: What is the purpose of the formative feedback techniques for programming languages learning?
- RQ2: What is the nature of the formative feedback techniques for programming languages learning?
- RQ3: Does the assessment method prioritize helping novices and influencing their programming learning?

Table 1 Key Search Terms.

Key Concepts	Search Terms
Computer Programming	"novice computer programmers" or "computer programming" or "programming skills" or "first year computer programming" or "programming concepts" or "introductory programming" or "automated system" or "Automated assessment"
Formative Assessment	"formative feedback" or "error correction" or "learner confidence" or "learner efficacy" or "adaptive assessment" or "computer programming assessment system"

3 Research Methodology

More study, according to [14], is necessary to fully understand the inquiry process (formative assessment) that takes place in a programming learning environment. In order to better comprehend the current state of research, this study will examine recent articles that were released between 2013 and 2023. This study used this time frame of the last 10 years because the review topic is only applicable to contemporary studies [37]. This research was guided by the following PICO question: "The impact of formative feedback to motivate novices in learning introductory programming". The What Works Clearinghouse Procedures and Standards Handbook, Version 4.0, from the U.S. Department of Education's Institute of Education Sciences served as the process [33]. The five stages were as follows: (a) creating the review methodology; (b) locating pertinent literature; (c) screening studies; (d) reviewing articles; and (e) reporting findings. This study follows these stages to systematically review the literature.

3.1 Data Sources and Search Strategies

The terms "Formative assessment" and "Computer programming" were broadly entered into databases using the Title, Keyword, and Abstract search functions to look for published publications between the years 2013 and 2023. Table 1 shows the alternative terms for the key terms in the search. Academic Search Complete, ERIC Library, ACM, IEEE and Science Direct were the databases combined. This research used search engines like Google Scholar, ResearchGate, and Academia to find manuscripts [22]. Additionally, core conferences in computer education such as SIGCSE, CompEd, ITiCSE, UKICER and ICPEC are taken into account. 197 articles were found in the initial search results since 1998. Based on the inclusion and exclusion criteria, these papers were evaluated at the title, abstract, and full text levels. 17 articles were produced as a result out of 22, and they were coded for the systematic review. As they concentrated on basic computer science, summative assessment, or gaming techniques rather than clearly focusing on introductory programming, formative assessment, or evaluation approaches, 5 articles were eliminated [51, 54, 43, 21, 15].

3.2 Inclusion and Exclusion Criteria

Each study must meet these screening requirements to be considered for this systematic review: Computer programming and formative assessment are the article's primary foci, and its publication dates range from 2013 to 2023. Its publication type is original research from peer-reviewed journals, and its research methodology includes both quantitative and qualitative approaches with a clear methods section and results presentation. Its language

Table 2 Inclusion and Exclusion Criteria.

Criterion	**Inclusion**	**Exclusion**
Timeframe	2013–2023	Prior to 2013 the focus in literature prior to this time was primarily on plagiarism.
Language	English	Non-English
Access	Full-text availability only	Only titles or abstracts available
Sample	Novice programmers	Programming languages, Web design, model development, advanced programming, visual or scratch programming
Type of publication	Peer-reviewed, original research, conference papers	Content that was not peer-reviewed or original
Focus of literature	Presenting findings that help design an enhanced formative assessment system for novice programmers (what works and doesn't work, and why).	Studies that present findings on summative assessment. Systems designed for gifted programmers.

is English, and its emphasis on formative assessments that are being used to support their programming learning. If a research study did not satisfy one or more of the inclusion requirements that shows in Table 2, it was excluded.

4 Data Coding

Content analysis was used to find categorical themes from narrative data used to inform research focus and feedback strategy. In order to draw logical conclusions, content analysis aims to organize and interpret the data collected [6, 26]. The data gathering method across all research was formative or automated assessment. All studies took place in either a higher education or secondary context. All of the research was carried out globally, primarily in Northern America and Europe.

4.1 Formative Feedback Purpose: (RQ1)

Using the main subcategories of formative feedback found in the chosen publications, authors expand the following categories as they are interrelated in achieving learning goals [1].

4.1.1 Scaffold

The term "scaffolding" describes how an assessment plan might lead students through the steps of a larger project, with the teacher acting as an experienced leader who offers advice along the way [44]. The design of an assessment can scaffold the steps of a larger project by asking students to complete the steps so they can receive formative feedback in between [19].

4.1.2 Motivation

A student's motivation is defined as their "willingness, need, desire, and necessity to engage in and be successful in the learning process" [52]. The two main categories of motivational factors are intrinsic and extrinsic [2]. Self-motivation is another name for intrinsic motivation, which is the strong desire to learn a subject. Extrinsic motivation occurs when actions are

Table 3 Purpose of formative feedback.

Instructional strategies	#	Studies
Scaffolding	5	[17][5][45][8][29]
Motivation	6	[19][38][4][3][23][31]
Engagement	4	[18][7][11][13]
Metacognitive/ Self-efficacy	2	[29][49]

taken to satiate an outside demand or receive an outside-imposed reward. Instead of just enjoying the activity or engaging in it for its own sake, they may be performed for their instrumental benefit.

4.1.3 Metacognitive and Self-efficacy

Understanding how to learn, participating actively, and reflecting on that engagement is defined as metacognition [10]. An individual's belief in their own ability to do well is known as self-efficacy [2]. For successful learning programming, metacognition and self-efficacy are crucial abilities [30].

4.1.4 Engagement

Engagement promotes learning and forecasts students' success [1]. It is a multidimensional meta-construct with elements of behavior, emotion, and cognition. Formative assessment helps students become more engaged in their learning and that makes them more confident in the subject [47].

4.2 Nature of Formative Feedback: (RQ2)

Authors distinguish between the studies using feedback techniques tailored to engage students in learning programming as follows:

4.2.1 Assessment Approach

This defines the assessment approaches to generate the feedback such as, self-assessment, peer assessment, (semi-) automated assessment on programming assignment.

4.2.2 Feedback Types

It defines what type of feedback the tool or system provides such as, standard error messages, customised error messages, testing report or grades.

4.2.3 Feedback Mechanism

It defines how the feedback is generated to notify such as unit testing, test cases, verification and validation, guided instructions or directions of tests.

Table 4 Nature of formative feedback.

#	Studies	Assessment approaches	Feedback types	Feedback Mechanisms
4	[5, 13, 17, 4]	Automatic	Test case reports w/grade	Automated testing
1	[18]	Automatic	Unit test results	Automated testing
2	[3, 11]	Automatic	Verification report	(Static) Verifier generated
2	[31, 53]	Peer	Customised error messages	Peer (Lecturer/ fellow) feedback
1	[29]	Self	Customised error messages	Rubric-based
3	[19, 38, 8]	Automatic	Standard error messages	Compiler
1	[49]	Peer	Customised error messages	Guided inquiry
2	[23, 7]	Semi-Automatic	Customised error messages	Feedback about code quality
1	[45]	Semi-Automatic	Customised error messages	Manual feedback

5 Results

5.1 Purpose of Formative Feedback (RQ1)

In order to determine the aim and different kinds of formative feedback strategies for learning programming languages, authors examined the relevant research studies in order to get the answers to the research questions. Overall, the majority of studies found that motivation, scaffolding, and engagement were the purposes of feedback, with metacognitive purposes being recognized in fewer studies (see Table 3). In general, the goal of all these studies is to inspire students who are learning programming at various levels.

5.2 Feedback Strategies (RQ2)

Automated assessment was more commonly employed to generate feedback rather of using other approaches. Customised error messages were provided for facilitating formative feedback that were most frequently used in addition to automated testing results and compiler-based standard error messages. They were customised by peer, manual feedback and rubric-based feedback. Guided inquiry, feedback on the quality of the code, and chatbot interaction are among the assessment mechanisms (see Table 4). In general, most systems offer customized error messages to help students comprehend the errors they committed.

5.3 Novices' Support (RQ3)

All of these studies generally aim to motivate students learning programming at different stages. This analysis found that, with the exception of one, no studies have focused especially on novices [45]. The assessment system that aids novices, however, makes use of Teaching Assistants' (TA's) feedback rather than automatic feedback [45]. As a result, no automatic formative feedback is primarily emphasizing novices to encourage their learning programming.

6 Discussion

Formative assessment systems offer automatic formative feedback to students who submit their solutions or programs. These systems use various technologies to provide this feedback. Students can receive automatic formative feedback based on automated unit and system tests in this system if the code is valid [18]; otherwise, they receive error messages so they can change it. This system tested how well formative feedback connects with students of diverse backgrounds and showed how it facilitated their learning of programming. Another system [17] employed laboratory exercises were accompanied by automated test cases created by the lecturer using solely free software testing tools that match industry standards in order to provide students with formative feedback as they were working. The immediate feedback and continued efforts to close the performance gap between actual and expected performance were lauded by the students, and the efficacy was determined to be successful [17].

The autoCOREctor, a tool for automated student-centered assessment, was created to be easily connected with learning management systems (LMSs)[5]. It encourages the development of a problem-statement scaffold for programming assignments and a straightforward test set with test cases to assist teachers in using it in diverse situations. Regarding the codes they entered, which they must attempt several times to pass, students receive grades and feedback. Because of this, the autoCOREctor's feedback was helpful, easy to understand, helped students improve their assignments, and increased their motivation. The work's drawback was that it was unclear whether it would be helpful for novice programmers or the introductory programming module.

A study looked at Algo+ and EPFL, two automated evaluation methods for online introductory programming courses [8]. Based on the discrepancy between the supplied program and the referent solution that is the closest match, Algo+ delivers comments. This distinction clarifies to the learner the processes to follow in order to arrive at the correct response. The feedback given by the EPFL grader is based on test cases and a check style process. The generated feedback educates students about the test case successes and failures by displaying the program's result and the anticipated output. The student's understanding of why the programs don't provide the right responses even though they are syntactically correct is improved by the feedback on the most well-liked incorrect programs.

Incorporating online coding tasks into formative assessment is examined in an article to determine its practicality and efficacy [4]. Positive results from the experimental investigation back up the use of online coding environments in introductory programming and algorithm courses. It argues that formative rather than summative assessments enhance the learning process for students. As tutoring systems, chatbots have been used in a variety of settings. A system aims to use chatbots to teach basic CS concepts while increasing work completion and engagement among students, especially female students [7]. Because they produce formative feedback immediately at the task level and use the input to guide students toward learning programming, chatbots are useful tools for formative feedback.

Another formative assessments technique is that formative evaluation is combined with automatic source code verification and validation feedback [3]. With the use of this system, formative assessments will be able to provide feedback on the verification outcomes. When errors are discovered during the automatic verification phase, students will be given a report, enabling them to both fix the errors and gain a deeper understanding of the code. Similar system evaluated a library for automated assessments created especially for static analysis [11]. The lecturer can personalize exercises, reuse verification, and modify the lesson for each student using the library. It showed how flexible feedback on verification helps students identify inefficient and incorrect code fragments and encourages them to adopt good programming techniques.

Guided inquiry learning (GIL), an illustration of an inductive collaborative learning strategy. Students are expected to complete the learning objectives and provide their peers and the instructor comments on the problem [31]. It showed how receiving peer feedback improved their programming skills. Another system that used formative evaluation based on peer code review caused students' programming abilities to consistently improve [53]. Peer code review and inspection is an effective strategy to ensure the high quality of a software by methodically examining the source code. With the use of peer feedback, the students were able to identify and correct their errors. Similarly, a study looked at how students in introductory courses who are not majoring in computer science respond to evaluation situations [45]. The automatic evaluation system was utilized with TA's support. Because of this, the manual's (TA's) input was useful but not always practical to access.

A framework for understanding the what, why, and how of formative assessment of inroductory programming in K–12 computer science was developed in order to answer the overall need for understanding formative assessment [19]. Thus, CS research on assessment design and programming learning, particularly student misconceptions, has an impact on the formative assessment questions' design [19]. Another study focused on how repeat questions can give students rapid feedback by using an internet platform called HERA [38]. It argues that the formative feedback was important and helpful for computational thinking [38]. According to another study [23], suggestions about unit length, unit complexity, and code duplication were the most beneficial to students. While this feedback does not help with the assignment, it enhances understanding [23].

A study [29] looked at the role of self-assessment in computer programming. Interest in learning is developing as a result of self-efficacy. The results of this study indicate that formative self-assessment may improve students' performance in an introductory programming course. Exercises in self-assessment with a rubric might be beneficial for first-year students. It was found that students who got comprehensive feedback on their learning were more motivated than those who merely got a rubric-based evaluation. According to this study, however, it was discovered that the self- assessment intervention had a practically significant impact on students' performance on programming projects. Another study [49] looked at the effects of open-ended assessment on students learning introductory programming in terms of performance and self-efficacy. Students who routinely completed the open-ended versions had higher average self-efficacy scores and assignment marks, though not by a statistically significant amount.

Because they can accommodate an endless number of students and submissions, Automated Testing and Feedback (ATF) systems were examined in this study to meet the demand [13]. The learning process can be completed by the student by submitting a novel answer and promptly receiving feedback. Feedback can address syntax errors, output accuracy, code performance, and if the code adheres to instructions exactly. The engagement and learning behaviors of learners in massive open online course (MOOCs) are examined in this study. This study found that code feedback is one of the most crucial aspects of MOOCs for programming and that there might be a positive trend toward ATF users getting better grades.

7 Conclusion

This systematic review revealed that most systems used customised feedback for formative assessment which adds more scaffolding to support learners' progression to the next level [13, 19]. In these systems, if the code is correct, students can receive automatic formative feedback based on automated unit and system tests; if not, they receive error messages to fix it out [18, 5]. Students praised the quick feedback and the ongoing drive to decrease the performance gap between actual and intended performance, and the efficacy was rated successful [17]. The study's positive findings support the use of formative assessments for introductory programming courses, and it makes the case that formative rather than summative assessments improve student experience [4]. Verifier generated feedback enabled the students to recognize and fix the errors using verification techniques [11, 3]. Students' programming skills steadily increased because of formative assessment based on peer code review [53, 31]. When an automated assessment system was incorporated with manual feedback, the result was more beneficial, but it was not always realistically accessible [45]. Self-efficacy has attracted growing interest in learning. The findings of this study suggest that formative self-assessment may enhance students' performance in a course on basic programming [49]. It was discovered that students who received granular feedback during their learning were more motivated than those who only received evaluation using a rubric and open-ended questions [29]. The positive is all these studies found the formative feedback (customised or standard error messages) were helpful to motivate, scaffold, engage or self-assess the learners in learning programming [23, 7, 29, 49, 45].

7.1 Limitations

However, only a few studies focused on novice learners and introductory programming [29, 45, 8]. There is no clear evidence that these formative assessments were helpful in motivating specifically the novice learners in programming assignments. By showing the result of the program that was submitted and the anticipated output, the generated feedback instructs students about the test case's success or failure [8]. The work's limitation was that it did not say whether it was useful for novice programmers who were just starting out or the introductory programming module [5]. Another limitation is that all these systems assess the same questions to all students. It does not support students with different abilities. Adaptive techniques are used in formative assessment to achieve its goals [56]. When a student provides an erroneous response to a question, the system can progressively lead the student through a discovery process that results in the proper solution, breaking down complex concepts one step at a time [35]. When students fail, it aids in providing assistance and encourages personalised learning [59].

7.2 Implications

This study provides an overall analysis of the formative assessment that underpins the programming module in various educational settings. For several factors, including scaffold, motivation, self-confidence, and engagement, enormous amounts of evidence was discovered. The utilization of feedback techniques and student participation in formative assessment, we discovered, had an impact. The findings of this review also suggest that several variables may have an impact on the various formative assessment strategies. To better assist novice programmers in learning programming, formative assessment needs to improve how it presents error messages. We could not find any research that addressed formative assessment or the use of feedback, despite the fact that these elements are probably crucial for inspiring novice learners. There is also less support for several criteria including adaptive strategy, purely because fewer research have looked into them. As a conclusion, this study recommends on how to use an adaptive strategy in the process of formative assessment, in order to especially motivate novices and boost their knowledge and confidence. Therefore, this study's next work will design a formative assessment system that uses the adaptive strategy with enhanced error messages to evaluate the ability level of the novice students and motivate them to learn programming to acquire the necessary knowledge.

References

1 Elizabeth Acosta-Gonzaga and Aldo Ramirez-Arellano. Scaffolding Matters? Investigating Its Role in Motivation, Engagement and Learning Achievements in Higher Education. *Sustainability*, 14(20), 2022. doi:10.3390/su142013419.

2 Francisca A. Adamopoulos. *Learning Programming, Student Motivation*, pages 1–10. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-319-60013-0_182-1.

3 Felipe I. Anfurrutia, Ainhoa Álvarez, Mikel Larrañaga, and Juan-Miguel López-Gil. Integrating Formative Feedback in Introductory Programming Modules. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 13(1):3–10, 2018. doi:10.1109/RITA.2018.2801898.

4 Dhakshina Moorthy Anitha and Dhakshina Moorthy Kavitha. Online coding event as a formative assessment tool in introductory programming and algorithmic courses —A exploration study. *Computer Applications in Engineering Education*, 28:1580–1590, 2020.

5 Enrique Barra, Sonsoles López-Pernas, Álvaro Alonso, Juan Fernando Sánchez-Rada, Aldo Gordillo, and Juan Quemada. Automated Assessment in Programming Courses: A Case Study during the COVID-19 Era. *Sustainability*, 12(18), 2020. doi:10.3390/su12187451.

6 Mariette Bengtsson. How to plan and perform a qualitative study using content analysis. *NursingPlus Open*, 2:8–14, 2016. doi:10.1016/j.npls.2016.01.001.

7 Luciana Benotti, Mara Cecilia Martnez, and Fernando Schapachnik. A Tool for Introducing Computer Science with Automatic Formative Assessment. *IEEE Transactions on Learning Technologies*, 11(2):179–192, 2018. doi:10.1109/TLT.2017.2682084.

8 Anis Bey, Patrick Jermann, and Pierre Dillenbourg. A Comparison between Two Automatic Assessment Approaches for Programming: An Empirical Study on MOOCs. *Journal of Educational Technology & Society*, 21(2):259–272, 2018. URL: http://www.jstor.org/stable/26388406.

9 Sébastien Combéfis. Automated Code Assessment for Education: Review, Classification and Perspectives on Techniques and Tools. *Software*, 1(1):3–30, 2022. doi:10.3390/software1010002.

10 David T. Conley and Elizabeth M. French. Student Ownership of Learning as a Key Component of College Readiness. *American Behavioral Scientist*, 58(8):1018–1034, 2014. doi:10.1177/0002764213515232.

11 Pedro Delgado-Pérez and Inmaculada Medina-Bulo. Customizable and scalable automated assessment of C/C++ programming assignments. *Computer Applications in Engineering Education*, 28:1449–1466, 2020.

12 Suzanne W. Dietrich, Don Goelman, Jennifer Broatch, Sharon M. Crook, Becky Ball, Kimberly Kobojek, and Jennifer Ortiz. Using Formative Assessment for Improving Pedagogy: Reflections on Feedback Informing Database Visualizations. *ACM Inroads*, 11(4):27–34, November 2020. doi:10.1145/3430766.

13 Hagit Gabbay and Anat Cohen. Investigating the effect of Automated Feedback on learning behavior in MOOCs for programming. In *Proceedings of the 15th International Conference on Educational Data Mining*, pages 376–383. International Educational Data Mining Society, July 2022. doi:10.5281/zenodo.6853125.

14 Donn Randy Garrison. *E-learning in the 21st century: A community of inquiry framework for research and practice.* Third Edition, October 2016. doi:10.4324/9781315667263.

15 Ashok Goel and David Joyner. Formative Assessment and Implicit Feedback in Online Learning. In *Proceedings of Learning with MOOCs III, Philadelphia, PA*, 2016. URL: https://www.davidjoyner.net/blog/formative-assessment-and-implicit-feedback-in-online-learning/.

16 Rubén González. ICE: An Automated Tool for Teaching Advanced C Programming. *International Association for Development of the Information Society*, 2017.

17 Peadar F. Grant. Formative test-driven development for programming practicals. *AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education*, 9, 2017.

18 Beate Grawemeyer, John Halloran, Matthew England, and David Croft. Feedback and Engagement on an Introductory Programming Module. *CoRR*, abs/2201.01240, 2022. `arXiv:2201.01240`.

19 Shuchi Grover. Toward A Framework for Formative Assessment of Conceptual Learning in K-12 Computer Science Classrooms. In *Association for Computing Machinery*, SIGCSE '21, pages 31–37, New York, NY, USA, 2021. `doi:10.1145/3408877.3432460`.

20 Sugandha Gupta and Anamika Gupta. E-Assessment Tools for Programming Languages: A Review. In *International Conference on Information Technology and Knowledge Management*, 2018.

21 Thomas Hainey, Gavin Baxter, Julie Black, Kenneth Yorke, Julius Bernikas, Natalia Chrzanowska, and Fraser McAulay. Serious games as innovative formative assessment tools for programming in higher education. In *ECGBL, 16th European Conference on Games Based Learning, 6 - 7 October 2022, Lisbon, Portugal*, June 2022. URL: `https://www.academic-conferences.org/conferences/ecgbl/`.

22 Jared Howland, Thomas Wright, Rebecca Boughan, and Brian Roberts. How Scholarly Is Google Scholar? A Comparison to Library Databases. *College & Research Libraries*, 70:227–234, May 2009. `doi:10.5860/crl.70.3.227`.

23 Julian Jansen, Ana Oprescu, and Magiel Bruntink. The impact of automated code quality feedback in programming education. In *Post-proceedings of the Tenth Seminar on Advanced Techniques and Tools for Software Evolution (SATToSE)*, volume 210, 2017.

24 Maria Kallia. Assessment in Computer Science courses: A Literature Review. In *King's College London*, 2017.

25 Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Transactions on Computing Education*, 19:1–43, September 2018. `doi:10.1145/3231711`.

26 Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. *Keele University and University of Durham technical report*, 2, January 2007.

27 Melisa Koorsse, Charmain Cilliers, and André P. Calitz. Programming assistance tools to support the learning of IT programming in South African secondary schools. *Comput. Educ.*, 82:162–178, 2015.

28 Nguyen-Thinh Le. A Classification of Adaptive Feedback in Educational Systems for Programming. *Systems*, 4(2), 2016. `doi:10.3390/systems4020022`.

29 Alex Lishinski and Aman Yadav. Self-evaluation Interventions: Impact on Self-efficacy and Performance in Introductory Programming. *ACM Transactions on Computing Education (TOCE)*, 21:1–28, 2021.

30 Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Trans. Comput. Educ.*, 22(4), September 2022. `doi:10.1145/3487050`.

31 Jose Manappattukunnel Lukose and Kuttickattu John Mammen. Enhancing Academic Achievement in an Introductory Computer Programming Course through the Implementation of Guided Inquiry-Based Learning and Teaching. In *Asia-Pacific Forum on Science Learning and Teaching*, volume 19,2, 2018.

32 Andrew Luxton-Reilly, Brett A. Becker, Yingjun Cao, Roger McDermott, Claudio Mirolo, Andreas Mühling, Andrew Petersen, Kate Sanders, Simon, and Jacqueline Whalley. Developing Assessments to Determine Mastery of Programming Fundamentals. In *Association for Computing Machinery*, ITiCSE-WGR '17, pages 47–69, New York, NY, USA, 2018. `doi:10.1145/3174781.3174784`.

33 Daniel M. Maggin, Erin Barton, Brian Reichow, Kathleen Lynne Lane, and Karrie A. Shogren. Commentary on the What Works Clearinghouse Standards and Procedures Handbook (v. 4.1) for the Review of Single-Case Research. *Remedial and Special Education*, 43(6):421–433, 2022. doi:10.1177/07419325211051317.

34 Sohail Iqbal Malik and Jo Coldwell-Neilson. A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 22:1089–1120, 2017.

35 Marina Marchisio, Tiziana Margaria, and Matteo Sacchet. Automatic Formative Assessment in Computer Science: Guidance to Model-Driven Design. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 201–206, 2020.

36 Lauren E. Margulieux, Briana B. Morrison, and Adrienne Decker. Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples. *International Journal of STEM Education*, 7:1–16, 2020.

37 Timothy Meline. Selecting Studies for Systemic Review: Inclusion and Exclusion Criteria. *Contemporary Issues in Communication Science and Disorders*, 33(Spring):21–27, 2006. doi:10.1044/cicsd_33_S_21.

38 Laura Orozco-Garcia, Carolina Gonzalez, Juan Montano, Cristian Mondragon, and Hendrys Tobar-Munoz. A Formative Assessment Tool to Support Computational Thinking in the Classroom. In *2019 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 185–188, 2019. doi:10.1109/ICVRV47840.2019.00043.

39 Elena C. Papanastasiou. *Adaptive Assessment*, pages 18–19. Springer Verlag, 2015.

40 Raymond Pettit, J.D. Homer, K.M. Holcomb, N. Simone, and Susan Mengel. Are automated assessment tools helpful in programming courses? *ASEE Annual Conference and Exposition, Conference Proceedings*, 122, January 2015.

41 James E. Prather. Beyond Automated Assessment: Building Metacognitive Awareness in Novice Programmers in CS1. In *Nova Southeastern University*, 2018.

42 Yizhou Qian and James Lehman. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ.*, 18(1), October 2017. doi:10.1145/3077618.

43 Yizhou Qian and James Lehman. Using an automated assessment tool to explore difficulties of middle school students in introductory programming. *Journal of Research on Technology in Education*, 54:1–17, January 2021. doi:10.1080/15391523.2020.1865220.

44 Brian J. Reiser and Iris Tabak. *Scaffolding*, pages 44–62. Cambridge University Press, United Kingdom, January 2014. doi:10.1017/CBO9781139519526.005.

45 Emma Riese and Stefan Stenbom. Experiences of Assessment in Introductory Programming From the Perspective of NonComputer Science Majors. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE Press, 2020. doi:10.1109/FIE44824.2020.9274060.

46 Siti Nurulain Mohd Rum and Maizatul Akmar Binti Ismail. Metacognitive Support Accelerates Computer Assisted Learning for Novice Programmers. *J. Educ. Technol. Soc.*, 20:170–181, 2017.

47 G. W. Scott. Active engagement with assessment and feedback can improve group-work outcomes and boost student confidence. *Higher Education Pedagogies*, 2(1):1–13, 2017. doi:10.1080/23752696.2017.1307692.

48 Nicole Shanley, Florence Martin, Nicole Collins, Manuel Perez-Quinones, Lynn Ahlgrim-Delzell, David Pugalee, and Ellen Hart. Teaching Programming Online: Design, Facilitation and Assessment Strategies and Recommendations for High School Teachers. *TechTrends*, 66, April 2022. doi:10.1007/s11528-022-00724-x.

49 Sadia Sharmin, Daniel Zingaro, Lisa Zhang, and Clare Brett. Impact of Open-Ended Assignments on Student Self-Efficacy in CS1. In *CompEd '19: Proceedings of the ACM Conference on Global Computing Education*, pages 215–221, April 2019. doi:10.1145/3300115.3309532.

50 Valerie J. Shute. Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189, 2008. doi:10.3102/0034654307313795.

51 Judith Stanja, Wolfgang Gritz, Johannes Krugel, Anett Hoppe, and Sarah Dannemann. Formative assessment strategies for students' conceptions—The potential of learning analytics. *British Journal of Educational Technology*, 54(1):58–75, 2023. `doi:10.1111/bjet.13288`.

52 Ricarda Steinmayr, Anne F. Weidinger, Malte Schwinger, and Birgit Spinath. The Importance of Students' Motivation for Their Academic Achievement – Replicating and Extending Previous Findings. *Frontiers in Psychology*, 10, 2019. `doi:10.3389/fpsyg.2019.01730`.

53 Qing Sun, Ji Wu, Wenge Rong, and Wenbo Liu. Formative assessment of programming language learning based on peer code review: Implementation and experience report. *Tsinghua Science and Technology*, 24:423–434, August 2019. `doi:10.26599/TST.2018.9010109`.

54 Elise Trumbull and Andrea A. Lash. Understanding Formative Assessment Insights from Learning Theory and Measurement Theory. In *WestEd*, 2013.

55 Fabienne M. van der Kleij, Theodorus Johannes Hendrikus Maria Eggen, Caroline F. Timmers, and Bernard P. Veldkamp. Effects of feedback in a computer-based assessment for learning. *Comput. Educ.*, 58:263–272, 2012.

56 Jill-Jênn Vie, Fabrice Popineau, Éric Bruillard, and Yolaine Bourda. *A Review of Recent Advances in Adaptive Assessment*, volume 94, pages 113–142. Studies in Systems, Decision and Control, February 2017. `doi:10.1007/978-3-319-52977-6_4`.

57 Xiao-Ming Wang, Gwo-Jen Hwang, Zi-Yun Liang, and Hsiu-Ying Wang. Enhancing students' computer programming performances, critical thinking awareness and attitudes towards programming: An online peer assessment attempt. *Educational Technology and Society*, 20:58–68, January 2017.

58 Stelios Xinogalos, Tomáš Pitner, Miloš Savić, and Mirjana Ivanović. *First Programming Language in Introductory Programming Courses, Role of*, pages 1–11. Springer International Publishing, Cham, 2019. `doi:10.1007/978-3-319-60013-0_217-1`.

59 Albert Yang, Brendan Flanagan, and Hiroaki Ogata. Adaptive formative assessment system based on computerized adaptive testing and the learning memory cycle for personalized learning. *Computers and Education: Artificial Intelligence*, 3:100104, October 2022. `doi:10.1016/j.caeai.2022.100104`.