Emergent Issues in Developing an Automated Feedback System for Programming Assignment

Bolanle Abimbola
Dept. of Computer Science
Babcock University
Ilisan,Ogun State
Nigeria

Agbaje M.O.
Dept. of Computer Science
Babcock University
Ilisan,Ogun State
Nigeria

Akande Oyebola
Dept. of Computer Science
Babcock University
Ilisan,Ogun State
Nigeria

Izang A.A.
Dept. of Computer Science
Babcock University
Ilisan,Ogun State
Nigeria

ABSTRACT
Learning programming has been a painstaking task for many learners, with the traditional method of teaching and feedback providing limited assistance to students and minimal improvement in their skills. Automated feedback systems have been developed to improve programming education by using technology to analyze students' code, identify the mistakes or the areas for improvement and provide personalized feedback. This paper focuses on the potential advantages and disadvantages of deploying automated feedback systems in programming courses. We examine diverse automated feedback methods including static code analysis, test case evaluation, and intelligent tutoring systems. Furthermore, we look at the effect of these systems on student learning outcomes, participation, and motivation. Additionally, we make recommendations on how automated suggestions can be included in programming courses curriculum and indicate further research which is of vital importance in this changing area.

General Terms
Software Engineering, Programming Education, Personalized Feedback

Keywords
Automated Feedback, Learning , Programming

1. INTRODUCTION
Programming skills are of utmost importance in the era of technology and are more than essential in areas like computer science, software engineering, data science, or computational science. Effective programming education is key to the development of students' skills necessary for the realization of meeting the growing need for programmers and the creation of new technologies in the technical sphere, [1]. Nevertheless, getting into programming might turn out to be difficult, especially for beginners. Conventional forms of code instruction and elemental reviews, for instance, in-person code reviews and manual grading by tutors may take a lot of time involving the tutor, and they hardly offer immediate assistance and support [2]. In addition, traditional feedback methods become less and less scalable with an increase in class size.

Automated feedback systems have been identified as a way to solve problems and make programming education better. These systems perform a checking function using technology [3]. The system detects in the students' code submissions, errors, inefficiencies, or areas requiring them to improve on as they provide personalized feedback to help them improve their programming skills. Automated feedback can for instance be delivered in real-time which means that immediate assistance can be given during learning at the exact moment when it is necessary without this in any sense involving constant manual intervention from teachers. This paper looks at the possible advantages and difficulties of having the automation of the feedback system in programming classes. We provide an overview of specific techniques of automatic feedback available, for example, curly braces check-ups, test case evaluation, and tutoring systems which are intelligent. Additionally, we address the issue of the influence of automated feedback on students' learning results, activity, and motivation. Besides that, we suggest the best practices for the inclusion of the automated feedback into the programming curricula and we also propose the directions for future research in this field which is developing rapidly.

2. RELATED WORKS
2.1 Approaches to Automated Feedback
2.1.1. Static Code Analysis

Static code analysis is a formal process that does not require code execution but only analyzes its source code. This way we can use the scan to mark syntax errors, code style issues, and possible logical or semantic errors based on the predefined rules or patterns. Static code analysis tools might be used to give some syntax checks against such coding conventions as variable naming, code structure, potential security weaknesses and other criteria [4]. One of the benefits of static code analysis lies in its ability to detect problems in the consortium coding when it is running. This can assist students in identifying and correcting errors more effectively, which will in turn help them save time and effort. Besides that, this type of analysis can be embedded in the Integrated Development Environments (IDEs) or editors, which allows continuous learning of best practices as the students make their own code. On the other hand, static code analysis has its limits when it comes to catching runtime errors or assessing the functionality emulated by the program design. It relies on predetermined rules and patterns which may prove ineffective in the cases not of the pre-existing scenarios. In addition, the efficiency of static code analysis can be affected by the complexity of the programming language and the quality of the analysis rules.

2.1.2. Test Case Evaluation

Test case evaluation involves executing the student's code against a variety of predefined test cases and comparing the outcome with the anticipated results. Applying this strategy to program debugging can assist in error identification regarding functionality and evaluate the accuracy of program output. Test case evaluation can be very helpful when students are working on programming assignments or projects that involve solving

problems or developing algorithms[5]. It gives a clear indication of the functionality of a program. By showing students a comparison between the program's output and the expected results, they learn how to identify in which area their code differs from the wanted reaction. Furthermore, test case evaluation can be automated, which will enable for mass and quick feedback delivery.

On the other hand, test case evaluation is unable to identify problems which are caused due to the code structure, readability, or speed. The main objective is the proper functioning of the program without paying attention to other factors that can affect the programming, such as choosing the right algorithm or code style. Also, it is a time-consuming and challenging task to create comprehensive and representative test cases, especially for complex programming assignments [6]

2. 1. 3 Intelligent Tutoring Systems

Intelligent tutoring systems (ITS) are complex, supporting systems based on the principles of artificial intelligence, machine learning, and educational psychology, which provide students with individualized and self-adjusting feedback. The primary purpose of these systems is to build a model of the learner's current understanding, to find their strengths and weaknesses, and to offer specific feedback and tutorial support [7].ITS may utilize strategies such as knowledge tracing which captures a student's command level of individual skills or concepts and hence, it may tailor the feedback and instruction in a way that suits the student's role. Besides, ITS can implement natural language processing to analyze and offer feedback on students' code comments or explanations, thus providing more holistic feedback on programming concepts and problem-solving strategies .On the list of ITS valuable features is the fact that they can deliver highly personalized and adaptive feedback that reflects the specific requirements of each learner and the rate of his progress. Through all these processes, the student's knowledge model will be made, and it will be updated as their performance improves. For the student model such education is a challenging task that necessitates a lot of research and development. These learning systems should not only combine the domain rules, learning strategies and artificial intelligence methods but should be capable of correct and useful feedback.

2.2 Benefits of automated feedback system

2.2.1 Immediate and Personalized Feedback

Automated systems have tailored feedback to many students which helps in improving their comprehension speed. Automatic feedback is a change of the past method of feedback which required the time taken for manual grading and scheduling as opposed to that when a student is working on a programming assignment or project [8]. Immediate feedback is key to effective learning because it allows students to discover and solve errors or mistakes immediately before these issues become fixed or result in more misconceptions. With the feedback given to them on the spot when they encounter a problem or work incorrectly, students can revise their performance, allowing for faster progress and correction of inadequate understanding [9].

Moreover, an automated feedback system can be used for personalized feedback using a student's source code submission or achievement as a starting point. This individualized feedback talks about individual errors, inefficiencies, or areas for improvement which will provide the student with targeted guidance and support based on his/her unique needs and learning progress.

2.2.2 Scalability and Efficiency

Quality and speed of feedback to students can be greatly improved upon through the implementation of automated systems. Automated feedback systems also help when dealing with large classes or online learning environments due to their scalability and efficiency. The ways by which feedback can be given by other students or by instructors apart from manual code reviews or grading becomes subjective, hence, the time required to carry them out grows along with the number of students [3]. The automated feedback system can analyze and provide feedback to a large number of code submissions quickly and efficiently without the need for manual intervention. This scalability allows mentors to give every student effective feedback in a much larger population than would be possible without automation, while dramatically reducing the workload and the overall time required for manual grading and feedback processes. By the same token, automatic feedback systems could integrate the delivery of feedback by removing repetition between teachers and students, hence, students receive their feedback continuously throughout their learning experience. This efficiency can be a factor of a more effective and interesting learning experience for the students as they will be able to get prompt advice and support when they need it.

2.2.3 Improved Learning Outcomes

Feedback systems with immediate, personalized, and consistent feedback between the personalized feedback and programming courses learning outcomes can enhance student learning outcomes in programming courses. Early feedback is really a detect and correct human mistakes since such problems are easier to trace and brush away. With that, students won't eventually be faced with bigger problems that can be more difficult to sort out. Timely and targeted feedback can enhance student performance and make them understand and memorize programming concepts and skills better [10]. Automated feedback systems are a tool to reinforce what is right in terms of the coding and to aid students develop a greater grasp of the theories of the programming principles for they must fully understand the theories besides knowing the implementation solely of code [3]. In addition, an automated feedback system may adapt delivery modes to multiple styles and preferences of learning, including providing alternative feedback options such as visual aids, the code embed options, or interactive examples. The diversity in the presentation of feedback can provide for different learning needs and therefore, will contribute to the overall effectiveness of the feedback process [11].

2.2.4 Increased Engagement and Motivation

Automated feedback could be a motivation boosting factor leading to enhanced student engagement in coding courses. The immediate and personalized feedback gives the students a chance to stay actively engaged with the learning process since after encountering challenges or errors prompt assistance and guidance is provided to them [12].

Timely feedback can help students to avoid discouragement and frustration, as they can quickly identify and solve problems, instead of facing roadblocks that may lead to disengagement or demotivation. Furthermore, the aid of such feedback systems can lead to a heightened sense of autonomy and independent learning. Students can get hints and directions without constantly demanding an instructor's presence[13]. Moreover, feedback systems could also build in gamification activities either represented in form of level up, achievements

or leaderboards to help with goal achievement and increase the desire to learn. Students may find the feeling of accomplishment by receiving constructive feedback on their progress and skills [14].

2.3 Challenges and Limitations of Automated Feedback Systems

2.3.1 Technical Complexities

Along with automatic feedback systems for programming instruction which raise their own set of technical issues such systems need to be able to read and analyze a variety of programming languages, as well as cope with different coding styles and conventions, and spot the errors, inefficiencies, or the opportunities for improvement. Indeed, A particularity consists of automated feedback systems which undergo complicated algorithms and heuristics by bursting out relevant feedback. Creating these algorithms and making sure that they work well and the same as before for any programming task is an enormous difficulty, and it requires doing tests and solving problems a lot of times [15].

Additionally, integrating automated feedback systems with existing programming tools, such as Furthermore, combining automated feedback system with existing programming tools like IDEs or learning management systems may have some technical barriers. It is very important to make sure that the software can be integrated and compatible with many different programs and environments so that the students can have a smooth and efficient feedback process.

2.3.2 Interpreting and Addressing Feedback

Although automatically given feedback systems are aimed to offer meaningful and clear guidance, they may put other students in a position of having to understand the provided feedback, which can be a challenge for some learners, especially those that are in the beginner stage. The interactions may be boring or utilizing technical wording that cannot be easily understood by students who do not have any programming background [16]. Besides, the feedback and the consideration of the proposed improvements may be linked to the need for a more profound understanding of computer programming concepts or problem-solving strategies. One of the obstacles that students may face is that the feedback may not be clear enough for them to turn into practical revisions and improvements in their code. Hence, students may end up getting frustrated or get stuck or they may lose their progress.

For this purpose, the need to give the students the requisite training as well as guidance on how to interpret and properly apply the automated feedback is vital. This may be done by incorporating the explanations or examples within the feedback system, providing additional tutorial materials, or introducing the possibility for the student to interact with the human instructor and get guidance.

2.3.3 Pedagogical Considerations

The introduction of automated feedback systems into programming education raises some pedagogical issues. Though these systems allow students to get instant feedback and guidance, it has to be ensured that they are following acceptable pedagogical principles and really help the learners acquire different skills.

One of the major points to keep in mind is the delicate balance between automated feedback and human interaction. Though the automation system can do a good job of helping students with specific assignments and feedback on programming tasks, humans are still the ones who play a major role in making sure that the students understand the big picture, are good at solving problems by themselves, and get the personal mentoring and training they need [17].The feedback given by the automated systems should be designed and structured in a way that promotes deep learning and not shallow and formulaic ways of programming. The feedback is required to induce critical thinking, problem-solving, as well as a conceptual understanding of programming principles rather than only normative syntax or code implementations [18].

2.3.4 Integration with Curricula

Introducing feedback systems in the existing curriculum of programming can be a daunting task. This is achieved by the skillful adjustment of the feedback system's capabilities and outputs to the learning objectives, course content, and instructional approaches of the curriculum. The process of making the integration with effectiveness is the teachers, curriculum developers, and the creators of the automatic feedback developers to be attached. The feedback system should be the one that is specifically made for the programming theories, problem-solving tactics, and the programming practices that are the subjects in the course[19]. Besides, the teachers might have to change their teaching techniques and course structure to be able to use the advantages of the automated feedback system. This might be that of redefining assignments, changing the assessment schemes, or introducing other new instructional strategies that would be in the line with the automated feedback which would be a part of the whole learning process.

3. BEST PRACTICES AND RECOMMENDATIONS

3.1 Combining Automated Feedback with Human Instruction

The integration of Automated Feedback with Human Instruction is a significant advancement in education and will indeed help to improve the learning experience of student. While the feedback systems that are automated have many advantages, it is important to understand their limits and the significance of combining them with human instruction and guidance. Automated systems should be deemed as tools to complement rather than fully depend on human instructors and their knowledge.

Human teachers are important in that they can provide individualized supervision, enhance comprehension of general concepts, and promote the development of critical thinking skills and solve unresolved issues. They can make the information given by the automated systems more realistic, explain what is unclear and give the students some more examples, and also answer the complex questions or the students' misunderstandings that may occur [17].

Through incorporating automated feedback systems together with human instruction, instructors will be able to draw on the strengths of both methods. Automatic systems can give immediate and widespread feedback on pinpoint programming jobs or assignments whilst human teachers can work on the next level of understanding, that is, personalized instruction and supporting students through the complex and more challenging tasks of the program.

3.2 Providing Meaningful and Actionable Feedback

In order to make feedback systems automatically efficient, the feedback should be meaningful, actionable and align well with

individual learning goals of students. The feedback should not only identify errors but should also be very direct and explicit in communicating how these errors may be resolved and avoided. The effective feedback should be constructive one, the one that will make students aware of their mistakes and offer them suggestions for improvement. In such an endeavor; it should be combined with the bigger topic which includes programming concepts and emphasis on problem-solving strategies which then should deepen the understanding of the underlying principles [20]. Also, the feedback ought to be shown in a language that is comprehensible and can be understood by least the variable learners. Through the use of unambiguous words, visual tools, and interactive examples, students can understand and use the feedback in their programming works.

3.3 Fostering a Supportive Learning Environment

Feedback systems via automation should be supplemented by engagement in building a friendly and cohesive atmosphere for student learning. Programming can be a difficult task, and students can encounter problems, obstacles, and feelings of failure, particularly when they get negative feedback. It is even more crucial that the feedback, either automated or person-based, is considered a chance for improvement and growth. Teachers and learning materials of the course should be in line with growth mindset since it is perfectly normal to make mistakes while learning something new and that sticking with it and learning from essential feedback are the keys to programming skills mastery [21].

Furthermore, when designing the collaborative and safe learning environment, educators might be able to curtail the new unfavorable effects. Through the promotion of peer support, organizing group discussions, and giving the students a chance to share their experiences and strategies, a community is created and a more positive attitude to feedback and continuous improvement is nurtured.

3.4 Continuously Evaluating and Improving Feedback Systems

The evaluation and continual improvement of automated feedback systems is imperative to guarantee their effectiveness and pedagogical relevance in a changing educational environment where students develop more digital skills. This is a process that requires regular examination of the quality and the influence of the feedback that is given, and also the collection of input from instructors and students on their experiences with the system.

Data-driven methods, including student performance and engagement metric analysis, can help to determine the performance of the feedback system. It is also obvious that they will help to determine areas of improvement. Another part of this research is getting qualitative feedback from the students who use the platform and their instructors so mistakes, problems and development areas will be revealed [22].

On the basis of these assessments, the feedback system will be improved upon and readjusted. Such could include the improvement of algorithms in providing more reliable and relevant suggestions, the enhancement of the user interface and how feedback is presented, and the integration of new functions and capabilities that fit the learning styles of students and engage them more with the platforms. Omitting an ongoing assessment and renovation of automated feedback tools becomes a prerequisite, not only for their ability to have a lasting impact on the progress of programming education but also for keeping the related field abreast of the latest trends.

3.5 Integration with Emerging Technologies

The innovation in technology is a continuous process, so further research should be done about the integration of automated feedback systems with new technologies that can make programming education better. Furthermore, the incorporation of virtual or augmented reality environments will deliver immersive and interactive learning opportunities for students as they will have a chance to merely understand the code and manipulate it in a new and fascinating way.

Furthermore, integration of natural and conversational AI assistants or chatbots will bring about a more natural way for students to interact with automated feedback systems. These AI assistants could be the ones to engage in conversational exchanges, giving feedback and guidance in a more human-like way, which could in turn improve the understanding and relevance of the feedback.

The integration of automated feedback systems with various online coding platforms, collaborative development environments, and cloud-based programming tools, is also a central concern. These integrations can help provide real-time feedback and collaboration allowing students to have useful support or guidance while working on coding projects or participating in coding challenges and hackathons.

Therefore, research should also look at the use of automated feedback systems in the context of the new programming paradigms such as low-code or no-code development environments that aim to make programming more accessible to non-technical users. The development of automated feedback systems, together with new paradigms, is able to widen the footprint and impact of educational programming which will help more learners to acquire coding skills and thus contribute to technology development.

4. CONCLUSION

Emerging technologies such as automated feedback systems provide a way forward in learner-centered programming education by making feedback more timely, personalized, and even scalable. Although these systems may involve some technical difficulties and pedagogical issues, the best practices like combining human and automated feedback can help to achieve maximum effect. As new technologies are being developed, focusing not only on intelligent tutoring systems, personalized learning pathways, and ethics will be important to grasp the maximum benefits of automated feedback in programming education. Although this adoption of cutting-edge technologies is not always easy for teachers to embrace, refining the use of these tools is one sure way of creating engaging and compelling learning experiences that prepare students for the "technology everywhere" era.

5. REFERENCES

[1] Zinovieva, I., Artemchuk, V., Iatsyshyn, A.V., Popov, O., Kovach, V., Iatsyshyn, A.V., Romanenko, Y., and Radchenko, O.: ‘The use of online coding platforms as additional distance tools in programming education’, in Editor (Ed.)^(Eds.): ‘Book The use of online coding platforms as additional distance tools in programming education’ (IOP Publishing, 2021, edn.), pp. 012029.

[2] Buhr, S.P.: ‘21st Century Physics Homework: A Mixed-Methods Approach Evaluating How an Individualized Online Homework Platform Can Provide Quality Feedback and Help Physics Students Engage in Self-Regulated Learning’, University of South Carolina, 2020.

[3] Messer, M., Brown, N.C., Kölling, M., and Shi, M.: 'Automated grading and feedback tools for programming education: A systematic review', ACM Transactions on Computing Education, 2024, 24, (1), pp. 1-43.

[4] Mehrpour, S., and LaToza, T.D.: 'Can static analysis tools find more defects? a qualitative study of design rule violations found by code review', Empirical Software Engineering, 2023, 28, (1), pp. 5.

[5] Paiva, J.C., Leal, J.P., and Figueira, Á.: 'Automated assessment in computer science education: A state-of-the-art review', ACM Transactions on Computing Education (TOCE), 2022, 22, (3), pp. 1-40.

[6] Pan, R., Bagherzadeh, M., Ghaleb, T.A., and Briand, L.: 'Test case selection and prioritization using machine learning: a systematic literature review', Empirical Software Engineering, 2022, 27, (2), pp. 29.

[7] Mousavinasab, E., Zarifsanaiey, N., R. Niakan Kalhori, S., Rakhshan, M., Keikha, L., and Ghazi Saeedi, M.: 'Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods', Interactive Learning Environments, 2021, 29, (1), pp. 142-163.

[8] Leinonen, J., Denny, P., and Whalley, J.: 'A comparison of immediate and scheduled feedback in introductory programming projects', in Editor (Ed.)^(Eds.): 'Book A comparison of immediate and scheduled feedback in introductory programming projects' (2022, edn.), pp. 885-891.

[9] Alibali, M.W., Brown, S.A., and Menendez, D.: 'Understanding strategy change: Contextual, individual, and metacognitive factors', Advances in child development and behavior, 2019, 56, pp. 227-256.

[10] Goodman, B.E., Barker, M.K., and Cooke, J.E.: 'Best practices in active and student-centered learning in physiology classes', Advances in Physiology Education, 2018, 42, (3), pp. 417-423.

[11] Tai, J., Ajjawi, R., Boud, D., Dawson, P., and Panadero, E.: 'Developing evaluative judgement: enabling students to make decisions about the quality of work', Higher education, 2018, 76, pp. 467-481.

[12] Almusaed, A., Almssad, A., Yitmen, I., and Homod, R.Z.: 'Enhancing student engagement: Harnessing "AIED"'s power in hybrid education—A review analysis', Education Sciences, 2023, 13, (7), pp. 632.

[13] Lee, S.M.: 'Factors affecting the quality of online learning in a task-based college course', Foreign Language Annals, 2022, 55, (1), pp. 116-134.

[14] Schunk, D.H.: 'Self-regulation of self-efficacy and attributions in academic settings': 'Self-regulation of learning and performance' (Routledge, 2023), pp. 75-99.

[15] Keuning, H., Jeuring, J., and Heeren, B.: 'A systematic literature review of automated feedback generation for programming exercises', ACM Transactions on Computing Education (TOCE), 2018, 19, (1), pp. 1-43.

[16] Lai, C., Hu, X., and Lyu, B.: 'Understanding the nature of learners' out-of-class language learning experience with technology', Computer assisted language learning, 2018, 31, (1-2), pp. 114-143.

[17] Keuning, H., Jeuring, J., and Heeren, B.: 'Towards a systematic review of automated feedback generation for programming exercises', in Editor (Ed.)^(Eds.): 'Book Towards a systematic review of automated feedback generation for programming exercises' (2016, edn.), pp. 41-46.

[18] Wang, X.-M., and Hwang, G.-J.: 'A problem posing-based practicing strategy for facilitating students' computer programming skills in the team-based learning mode', Educational Technology Research and Development, 2017, 65, pp. 1655-1671.

[19] Loksa, D., Ko, A.J., Jernigan, W., Oleson, A., Mendez, C.J., and Burnett, M.M.: 'Programming, problem solving, and self-awareness: Effects of explicit guidance', in Editor (Ed.)^(Eds.): 'Book Programming, problem solving, and self-awareness: Effects of explicit guidance' (2016, edn.), pp. 1449-1461.

[20] Renkl, A., and Atkinson, R.K.: 'Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective': 'Cognitive Load Theory' (Routledge, 2016), pp. 15-22.

[21] Zeeb, H., Ostertag, J., and Renkl, A.: 'Towards a growth mindset culture in the classroom: Implementation of a lesson-integrated mindset training', Education Research International, 2020, 2020, pp.1-13.

[22] Sentance, S., and Csizmadia, A.: 'Computing in the curriculum: Challenges and strategies from ateacher's perspective', Education and information technologies, 2017, 22, pp. 469-495.