# Automation Support for Giving Feedback in Learning Programming by Doing

Arthur Rump
University of Twente
Enschede, The Netherlands
arthur.rump@utwente.nl

## Abstract

Providing effective feedback on open-ended programming assignments is a challenge, particularly at scale. This challenge impacts student learning, as timely and accurate feedback is crucial to improve their programming and program design skills. We intend to develop an approach leveraging automation to support the feedback process, without requiring changes to assignments that enable full automation. This method aims to improve the consistency and efficiency of giving feedback, helping students learn more effectively.

## CCS Concepts

• **Applied computing → Computer-assisted instruction**; • **Social and professional topics → Student assessment**; *Computer science education*; *Computer engineering education.*

## Keywords

automated assessment, automated feedback, programming education, open-ended assignments, learning-by-doing

**ACM Reference Format:**
Arthur Rump. 2025. Automation Support for Giving Feedback in Learning Programming by Doing . In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2025), June 27-July 2, 2025, Nijmegen, Netherlands.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3724389.3731300

## 1 Context and motivation

Learning to program requires practice, but that is not all: actually learning from practice requires receiving feedback. In programming education this is commonly achieved through automated feedback: in many introductory programming courses, exercises come with test cases to immediately tell a student whether they implemented a function correctly. This works well for small and strictly defined exercises, where students write a single function or a small program with strictly defined input/output relations. However, we don't just want to teach our students how to write a single function or a small program. If we want them to learn how to design and build more complex programs, the principle of constructive alignment [1] tells us that students should also practice with and be assessed on designing and building more complex programs.

Practising design skills requires more open-ended assignments where students can actually make design choices in their programs [8]. A key point of our open-ended assignments is that students are typically not required to adhere to a given interface; not on the method or class level, nor for the user interface. For some assignments students are free to determine the functionality of their programs, but even when functional requirements are given, these tend not to specify the required user interactions in detail.

These open-ended assignments result in more complex programs, which amplifies the challenge of giving effective and timely feedback, especially with a large group of students. There are currently few tools available that provide effective automation support in this context, so the current feedback practice relies on a large team of teaching assistants. While there are certain benefits to this approach, it does introduce a risk of inconsistent assessment and feedback [9].

As we are moving away from purely summative assessments towards more formative assessments throughout the course, the impact of these challenges increases. Our goal is to develop an automation-assisted workflow that supports teachers and teaching assistants in giving feedback, combining both automated and manual feedback. We are not trying to fully automate assessment of these assignments, but rather to support the work done by teachers and teaching assistants.

## 2 Background

Tools for automated assessment and feedback are a well-researched topic, especially in programming education [2]. Most of these tools, however, work well on small and strictly defined exercises that can be evaluated with simple unit tests [10]. Tools that apply techniques other than testing are often difficult to configure and hard to adapt to new exercises [6]. These limitations can disincentivize students from building creative or innovative solutions [4].

A constructivist view of education argues that open-ended assignments are necessary for learning, for example in the form of a *driving question* in project-based learning [7]. These assignments are sometimes called *ill-defined*, because there is no definite end result students are expected to produce [3]. In practice, some assessment criteria are more strictly defined than others, opening the door for at least partial automation [12].

There has been recent work that aims to combine manual and automated assessment, either by auto-grading some criteria and extracting relevant information for manual grading [5] or by using generative AI to provide suggested assessments [11]. These approaches typically aim to automate as much as possible and then let an assessor take over for manual assessment. While this is a good first step, more complex assignments could benefit from more complex interactions of automation and manual assessment.

This aligns with a need for customizability and flexibility, which has recently been found as one of the most-valued features in automated assessment tooling [14]. This includes compatibility with languages and frameworks used in a course, but also being able to tailor a tool to the unique assessment needs of that course.

## 3 Problem statement

Giving feedback on open-ended programming assignments is still largely a manual task, which can take a lot of time. When distributed across many people, which is necessary when there are many students in a course, there is a risk of inconsistent assessments and inconsistent feedback. Moving away from purely summative assessments towards more formative assessments throughout the course further exacerbates these challenges. The use of open-ended assignments is an important part of the learning experience, so the use of automation should not force changes in the assignments that come at the expense of this quality.

## 4 Research goals

Our aim is to develop a tool-assisted workflow that uses automation to support giving feedback on open-ended programming assignments. The main goals are to increase consistency and reliability of feedback and assessment and to increase the efficiency of providing this feedback to students. Full automation is an explicit non-goal, so one of the challenges is to find a workflow that effectively combines automated and manual feedback. Within the workflow, we want to ensure that constructive alignment, validity and transparency are at least maintained at their current levels.

## 5 Research methods

Our first focus is on developing a workflow that combines manual and automated assessment. Given the goal to improve consistency and efficiency, it makes sense to evaluate a prototype of this workflow empirically by comparing assessments performed using a traditional paper rubric with assessments performed using our prototype. The exact details of this study, like the dataset to use, are still to be determined.

Working from this prototype, we can then consider the usability of the workflow to give continuous formative feedback throughout a course. The evaluation of our designs on this front would be more focused on the experience of teaching assistants and students who are using the workflow.

## 6 Current and expected contributions

Our first contribution concerns the requirements of an automated assessment tool [13]. One of the conclusions from that study is that teachers are not looking for a tool that performs automatic grading, but would rather have a tool that supports them in the assessment process. The main contribution we aim to make is an empirically evaluated workflow and tool to support the assessment of open-ended programming assignments.

## References

[1] John Biggs and Catherine Tang. 2011. *Teaching for quality learning at university* (4th ed.). McGraw-Hill/Society for Research into Higher Education/Open University Press.

[2] Anderson Pinheiro Cavalcanti, Arthur Barbosa, Ruan Carvalho, Fred Freitas, Yi-Shan Tsai, Dragan Gašević, and Rafael Ferreira Mello. 2021. Automatic feedback in online learning environments: A systematic literature review. *Computers and Education: Artificial Intelligence* 2 (2021), 100027. doi:10.1016/j.caeai.2021.100027

[3] Philippe Fournier-Viger, Roger Nkambou, and Engelbert Mephu Nguifo. 2010. Building Intelligent Tutoring Systems for Ill-Defined Domains. In *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 81–101. doi:10.1007/978-3-642-14363-2_5

[4] Marcelo Guerra Hahn, Silvia Margarita Baldiris Navarro, Luis De La Fuente Valentin, and Daniel Burgos. 2021. A Systematic Review of the Effects of Automatic Scoring and Automatic Feedback in Educational Settings. *IEEE Access* 9 (2021), 108190–108198. doi:10.1109/access.2021.3100890

[5] Henry Hickman and Tim Bell. 2024. Automated Assessment: Does It Align With Teachers' Views?. In *Proceedings of the 19th WiPSCE Conference on Primary and Secondary Computing Education Research (WiPSCE '24, Vol. 10)*. ACM, 1–10. doi:10.1145/3677619.3678113

[6] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2019. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Transactions on Computing Education* 19, 1 (11 Jan. 2019), 1–43. doi:10.1145/3231711

[7] Joseph S. Krajcik and Namsoo Shin. 2014. Project-Based Learning. In *The Cambridge Handbook of the Learning Sciences*, R. Keith Sawyer (Ed.). Cambridge University Press, 275–297. doi:10.1017/cbo9781139519526.018

[8] Angelika Mader, Ansgar Fehnker, and Edwin Dertien. 2020. Tinkering in Informatics as Teaching Method. In *Proceedings of the 12th International Conference on Computer Supported Education*. SCITEPRESS - Science and Technology Publications. doi:10.5220/0009467304500457

[9] Marcus Messer, Neil Brown, Michael Kölling, and Miaojing Shi. 2025. How Consistent Are Humans When Grading Programming Assignments? doi:10.35542/osf.io/nd6qy_v2

[10] Marcus Messer, Neil C. C. Brown, Michael Kölling, and Miaojing Shi. 2024. Automated Grading and Feedback Tools for Programming Education: A Systematic Review. *ACM Transactions on Computing Education* 24, 1 (Feb. 2024), 1–43. doi:10.1145/3636515

[11] Goda Nagakalyani, Saurav Chaudhary, Varsha Apte, Ganesh Ramakrishnan, and Srikanth Tamilselvam. 2025. Design and Evaluation of an AI-Assisted Grading Tool for Introductory Programming Assignments: An Experience Report. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSE TS 2025)*. Association for Computing Machinery, New York, NY, USA, 805–811. doi:10.1145/3641554.3701913

[12] Arthur Rump and Vadim Zaytsev. 2022. A refined model of ill-definedness in project-based learning. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (MODELS '22)*. ACM, 115–122. doi:10.1145/3550356.3556505

[13] Arthur Rump, Vadim Zaytsev, and Angelika Mader. 2025. Requirements for an Automated Assessment Tool for Learning Programming by Doing. In *2025 IEEE 18th International Conference on Software Testing, Validation and Verification (ICST)*. IEEE.

[14] Barrett Ruth and John R. Hott. 2025. Auto-grading in Computing Education: Perceptions and Use. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025)*. ACM, 1008–1014. doi:10.1145/3641554.3701900