Supporting the Semi-Automatic Feedback Provisioning on Programming Assignments

Sven Strickroth[1][0000−0002−9647−300X] and Florian Holzinger[1]

Institute for Informatics, Ludwig-Maximilians-Universität München, Germany
sven.strickroth@ifi.lmu.de

Abstract. Feedback is important for learning, however, manual feedback provisioning is time and resource consuming. For programming education, various systems have been developed to automate recurring tasks or the whole feedback generation. As fully automated systems are not always a viable option, this paper investigates how teaching assistants can be supported in semi-automated assessment of programming assignments to give good feedback more easily. An existing semi-automated e-assessment system is extended with configurable feedback snippets as well as adaptively recommended feedback snippets based on feedback of similar submission in order to evaluate the effects on the grading and the feedback given by teaching assistants. The results indicate that such feedback snippets lead to more consistent and motivational feedback, can help finding mistakes, and have no impact on the awarded grades.

Keywords: Feedback Provisioning · Assessment · Usability.

1 Introduction

A central component of introductory programming courses in higher education are homework assignments in which students write small programs themselves or extend existing programs on a regular basis. As learning programming is often perceived as hard by novices, a timely, individual, explanatory, and motivational feedback is important [3, 8, 9]. Fully automated systems can provide instant and objective feedback [6], but they are still limited as all tests and feedback need to be pre-defined and not all aspects such as creativity or implementation quality can be assessed automatically [7]. Therefore, manual or semi-automated grading is often necessary. This requires a significant amount of time and effort that only scales with the number of teaching assistants. There is, however, usually a (monetary) limitation on how many assistants can be hired and having more teaching assistants is likely to introduce inhomogeneities in feedback and grading.

This paper investigates the research question of how to support teaching assistants in semi-automatic assessment so that they can provide consistent and motivating feedback more easily. The main idea is to automate recurring tasks and to provide pre-defined feedback snippets as well as adaptively recommended feedback snippets based on similar submissions that have already been assessed.

The remainder is organized as follows: First, the state of the art is presented. This is followed by descriptions of the prototype, the evaluation design, and the results. The paper closes with a discussion, a summary, and an outlook.

2 Exploring the Design Space: State of the Art

There is a large number of systems and approaches to support teachers in programming education in terms of manual and semi-automatic grading and providing feedback [6, 11]. All supportive systems collect the students' solutions at a central place and allow teaching assistants to download or view these.

There seem to be two (orthogonal) dimensions on which systems focus: First, there are semi-automated systems such as Praktomat [2] that automate recurring tasks, such as compiling and testing of submissions, and present the results for manual grading to the teaching assistants. Such automation has shown to be very helpful, e. g. to identify mistakes easily, and a (significant) time saver [10]. Second, there are systems such as Rubyric [1] that focus on supporting the manual grading and feedback provisioning by providing a (detailed) form/rubric. The scoring rubrics can be either static or configurable and some tools (e. g., Rubyric) allow teachers to define re-usable feedback snippets. Rubyric was found helpful by teaching assistants, feedback was more consistent, and students were satisfied with amount and quality of the feedback [1].

Based on the experience that novice programmers often make similar mistakes or have common misconceptions, there is a concept proposal to further support the feedback provisioning by an collaborative approach [4]: During the assessment of a submission, a system could adaptively suggest already entered feedback for similar submissions that can be reused and personalized in order to release teaching assistants from writing the same feedback repeatedly.

To summarize: In all systems known to the authors there is no system that supports both dimensions and also provides adaptive recommendations for feedback snippets. Also, the authors are not aware of any evaluation of the effects of such supportive features on the awarded grades and the provided feedback.

3 Adaptive Feedback Snippet Recommendations

The research gap identified in the previous section is addressed by an extension of the e-assessment system GATE, which is in use since 2011 [10]. GATE supports teaching assistants by displaying the results of tests that are automatically executed after the submission deadline, a teacher configurable rubric, and (optionally) with plagiarism detection by linking similar submissions.

For this research GATE was extended with teacher configurable feedback snippets and an adaptive feedback snippet recommendation that is based on a simple clustering approach [5]: The outputs of all failed automated tests are parsed to detect Java exceptions, failed unit test steps, or syntax errors. Then, the similarities to all other failed test outputs are calculated and stored in the database. Finally, during the assessment, already given feedback for similar errors

and test results is presented to the teaching assistants ordered by the calculated similarity. For technical reasons, the single-page assessment workflow needed to be split into a multi-page workflow so that the feedback snippets can be calculated for a specific assessment step (e. g. a syntax or unit test).

Fig. 1 depicts an excerpt from the GUI for the teaching assistants of the new interface. The assessment consists of three steps/pages (an overview of the steps can be seen at the upper right: for the first two steps automatic tests are linked and only the first one passed) and a separate overview page. In the middle there is the form for awarding grades and leaving feedback for the current step. In the figure it concerns the syntactically correctness and also shows the result of the automatic syntax test (PASSED, otherwise the error would be shown). Grades are automatically pre-filled when all tests of a step were passed. Additionally, misconceptions can be created or added to a submission (cf. [5, 4]). Directly below the form it is possible to select and insert a feedback snippet from a set of pre-defined or adaptively recommended feedback snippets. Always displayed at the bottom are the files submitted by the student (with syntax highlighting).

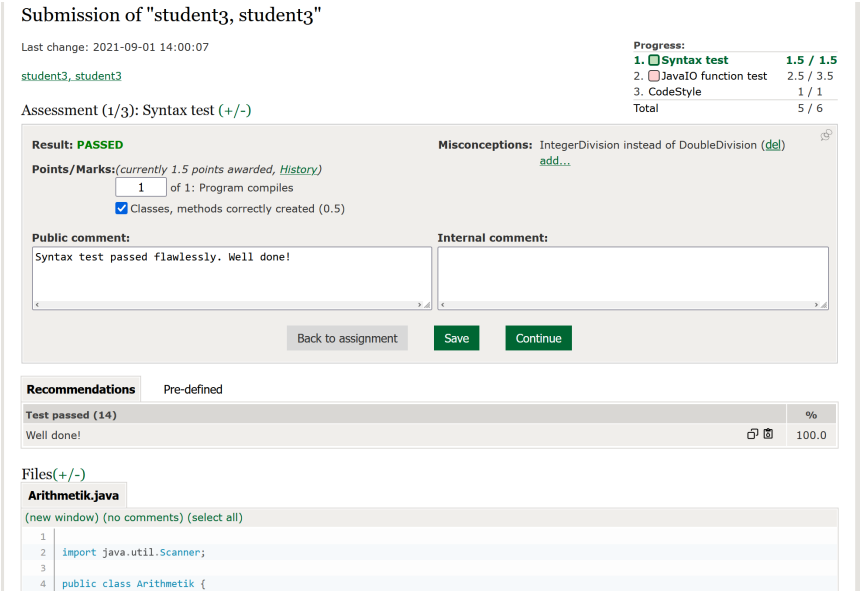

Fig. 1. User interface of the assessment interface of the prototype consisting of three assessment steps/pages, showing the first step *(translated to English)*

4 Evaluation

To investigate the effects on the grading and given feedback as well as the usability, an empirical lab study with a within-subjects design and a think-aloud

approach was conducted. Fifteen participants took part in the study (f=5, m=10) with an age between 23 and 55 (mean=ø=28.3, median=26). The participants were randomly split into two groups which either used the "old" system ($A \rightarrow B$) or the new prototype ($B \rightarrow A$) first to address ordering effects. All participants were either students (10) or teachers (5) from three German universities. Eleven participants already had experience as teaching assistants whereas four of these rated their teaching experience as little to medium and seven as much to very much. Nine participants were part of the teaching team for a bachelor's course on introductory programming at least once and three participants were part of the teaching team of programming practicals at least once. Two of the participants had experience with the GATE system and five have used other assessment tools.

For the study two mock courses each consisting of 20 students, the same programming assignment, and (comparable) submissions were constructed (based on real ones). For each course four submissions were not assessed, yet, of which two contained a mistake. During the study the participants got a quick introduction into the prototypes and then had the task to assess these eight submissions. The time needed for completing the assessment was measured. At the end a semi-structured interview was conducted. The study used Zoom with screen sharing and was recorded. Two 50 € Amazon vouchers were raffled among the subjects.

5 Results

All participants were able to use both systems to assess the eight submissions and it took approx. one hour per participant to finish the study. Except for one person (P12, $A \rightarrow B$) who did not spot a functional error with system A, all other participants detected all mistakes in the submissions to be assessed. However, P12 spotted a similar error with system B based on an adaptively recommended feedback snippet of an already assessed similar submission.

The average length of the feedback given with system A was 103 characters and 181 characters for system B – a statistically significant difference (p=0.001). An in-depth analysis for correct submissions shows that the given feedback is much shorter (A: ø=52, B: ø=158 characters) compared to incorrect submission (A: ø=133, B: ø=195) and that the given feedback is minimal longer in the first system used (ø=160) compared to the second system (ø=125). No other differences based on the order could be determined. Note, however, that there are two participants who once gave the very same comment in two steps. These two duplications have been resolved before calculating the quantitative statistics above. The qualitative analysis of the given feedback shows that in all comments the mistakes were clearly named except for the one participant (P12) who did not spot an error with system A. Independent of the system used for giving the feedback but dependent on the personal assessment style, some participants gave hints on how to resolve the mistake. Overall, the feedback given using system B seems to be more consistent/homogeneous and contains more positive/motivational comments compared to feedback given using system A (A: 23,

B: 48). The awarded grades show no (statistically significant) difference between the two systems (A: ø=4.7; B: ø=4.9, p=0.32; grades from 0 worst to 6 best).

The time required for the assessment of the submissions shows a statistically significant difference (p=0.048<0.05): The assessment with system A took 3:29 min. on average and with system B 4:13 min. (a difference of 21 %). Also, the assessment of correct submissions took about one minute (i. e. 33 %) longer with system B compared to system A. When system A was used first, the correction of incorrect submissions took about the same time (ø=4 min.) for both systems. When system B was used first, however, the assessment of incorrect submissions was about 44 % quicker with system A (A: ø=3:27, B: ø=4:59 min.). Correct submissions were always assessed quicker than incorrect ones.

In the interviews 80 % of the participants explicitly stated a preference of system B. System A was rated three times as not as supportive as system B. The feedback snippets were mentioned positively 7 times and were estimated to result into more consistent feedback by two participants. Three participants each stated that they felt they provided more positive feedback or more feedback overall with system B. However, two participants noted that the additional features also increase the overhead and lead to a longer time to get familiarized with system B. Five participants liked that the grades were pre-filled for steps in which all tests were passed and four participants rated the recommended feedback snippets as particularly helpful for spotting the cause of the mistakes. Additionally, three students missed a feature to directly correct a syntax error and being able to re-run the (unit) tests within the system.

Regarding the workflow, 12 participants rated the multi-step version as good but system A was classified as more efficient compared to system B by six participants. One participant (P6) explained this by saying everything is visible at a glance and this is particularly helpful as "assessment means multitasking". Three participants noted however that they were less afraid of making mistakes in system B, the feedback quality increases, the steps help structure the feedback, and allow to focus on one aspect at a time. – The detailed usability analysis and usage of the misconception feature are skipped here due to page restrictions.

6 Discussion, Conclusions and Outlook

This paper investigated how teaching assistants can be supported in semi-automatic assessment to give feedback more easily. A prototype was built that automates recurring tasks and provides pre-defined feedback snippets as well as adaptively recommended feedback snippets based on similar submissions. The results of an empirical lab study indicate that the feedback snippets have been very appreciated by the teaching assistants and lead to more consistent and motivational feedback without influencing the awarded grades. The snippets also seem to be particularly helpful for detecting causes of mistakes and do not result in many duplicate comments. The time needed for an assessment, however, seems to be higher for the prototype. This was particularly true for the first submissions – the last ones were assessed much quicker (about 40 %; 12 % for the old system).

Reasons could be that more clicks are needed for a single assessment but it is also possible that it is caused by the longer feedback or that the distinct steps triggered a deeper investigation of the submissions. Interestingly, the objective time measurement contradicts the subjective perception of five participants of being quicker with the prototype. Note, however, that the time measurement might be biased due to the think-aloud approach but should still show a tendency.

The main limitation of this research is the lab study with eight assignments to be assessed. This number might seem small, however, the time (one hour) per participant for the study was the limiting factor. Still, the participants came from different universities and cover a wide range of different experiences.

Pre-defined and adaptively recommended feedback snippets are likely to be transferable to (both formative and summative) assessments in STEM to improve the given feedback, even if automatic tests are not or only partially available, and might also be helpful in settings such as peer review and peer feedback.

The workflow needed to be changed from a single-page to a multi-page assessment for the adaptive feedback snippet recommendation. Several participants stated that they preferred either of the two workflows. Due to the study design no clear conclusions can be drawn upon the multi-page vs. single-page workflow. Therefore, further research is needed to compare both approaches with the same feature set and the procedures of how solutions are assessed.

References

1. Auvinen, T.: Rubyric. In: Proc. Koli Calling. ACM Press (2011). https://doi.org/10.1145/2094131.2094152
2. Breitner, J., Hecker, M., Snelting, G.: Der Grader Praktomat. In: Automatisierte Bewertung in der Programmierausbildung, pp. 159–172. Waxmann (2017)
3. Butler, M., Morgan, M.: Learning challenges faced by novice programming students studying high level and low feedback concepts. Proc. ascilite pp. 99–107 (2007)
4. Heller, N., Bry, F.: Human computation for learning and teaching or collaborative tracking of learners' misconceptions. In: Intelligent Systems and Learning Data Analytics in Online Education, pp. 323–343. Academic Press (2021)
5. Holzinger, F.: Kollaborative Unterstützung bei der semi-automatischen Bewertung von Programmieraufgaben. Master's thesis, LMU Munich, Germany (2021)
6. Keuning, H., Jeuring, J., Heeren, B.: A systematic literature review of automated feedback generation for programming exercises. TOCE **19**(1) (2018)
7. Laß, C., et al.: Stager: Simplifying the manual assessment of programming exercises. In: Proc. SEUH. CEUR-WS, vol. 2358, pp. 34–43 (2019)
8. Luxton-Reilly, A., et al.: Introductory programming: A systematic literature review. In: Proc. ITiCSE. pp. 55–106 (2018)
9. Moreno, R.: Decreasing cognitive load for novice students: Effects of explanatory versus corrective feedback in discovery-based multimedia. Instructional science **32**(1), 99–113 (2004). https://doi.org/10.1023/B:TRUC.0000021811.66966.1d
10. Strickroth, S., Olivier, H., Pinkwart, N.: Das GATE-System: Qualitätssteigerung durch Selbsttests für Studenten bei der Onlineabgabe von Übungsaufgaben? In: Proc. DeLFI. pp. 115–126. Bonn, Germany (2011)
11. Strickroth, S., Striewe, M.: Building a corpus of task-based grading and feedback systems for learning and teaching programming. iJEP **12**(5), 26–41 (2022). https://doi.org/10.3991/ijep.v12i5.31283