Designing Autograders For Novice Programmers

(A SIGCSE 2022 Workshop Proposal)

Chad Hogg*
Millersville University of Pennsylvania
Computer Science
Millersville, PA, USA
chad.hogg@millersville.edu

Maria Jump
Northeastern University
Khoury College of Computer Science
Boston, MA, USA
m.jump@northeastern.edu

*Corresponding Author

MODE

On-Site. Remote attendees could participate through videoconferencing.

ABSTRACT

Autograders have become an invaluable tool for instructors of computer programming courses. They not only ease the burden of manually grading many assignments, but more importantly provide students with a way to receive immediate feedback about their work and opportunities to revise and resubmit it. In the same way that compilers discover and report syntax errors in code, autograders can discover and report logic errors in code. But just as compiler messages are often indecipherable to novice programmers, autograder messages can be confusing and unhelpful to them. This workshop demonstrates strategies for designing autograders that will anticipate all of the mistakes that novice programmers might make and provide failure messages that are understandable to novice programmers.

SIGNIFICANCE AND RELEVANCE OF THE TOPIC

Many instructors of introductory programming courses have adopted the use of autograding software to examine student work and provide immediate feedback about the quality of their work. These sorts of tools have existed for many decades [2], but have become much more popular in recent years. A study of one particular proprietary autograding system showed five-fold growth in the last four years. [4] We are not aware of broader quantitative surveys, but informal conversations within the CS education community suggests that adoption is common. The 2021 Technical Symposium included 6 papers about the use of autograders.

There has been much work recognizing that the error messages provided by compilers often do not provide novice programmers with helpful information about what they have done poorly, though attempts to improve them have proven challenging. [3] [1]

Autograders do not (primarily) produce messages about syntax errors, but they must detect and notify the student about logic errors. This is difficult for two reasons. First, because the spectrum of errors that novice programmers may commit is vast, varied, and full of bizarre things that an experienced programmer would not think of. Second, because novice programmers have limited experience with debugging and technical jargon, and often only a very abstract sense of how computer programs work.

Designers of autograders that will be used by novice programmers thus have two responsibilities: to predict as many failure conditions as possible, and to produce informative feedback about each of those conditions. A naïve approach to writing an autograder, which simply runs the student program on some input and compares its output to what is expected is unlikely to do so. A more sophisticated approach might use unit testing, but this introduces additional failure modes (will the entire autograder fail if one unit to be tested is misnamed?) and can produce even less informative feedback if the test designer is not careful.

EXPERTISE OF PRESENTERS

One of the presenters has been teaching undergraduate computer science for 12 years, with at least one section of CS1 or CS2 most of those semesters. The other has been teaching introductory programming to graduate students who are new to CS for 4 years and taught CS1 and CS2 to undergraduates for 9 years before that. Both presenters began using autograders in their courses in 2016, and between the two of them they have experience with four frameworks: AutoLab, codePost, Gradescope, and Submitty.

ROUGH AGENDA FOR THE WORKSHOP

(1) Introductions (10 minutes)
(2) Workflow for developing / testing an autograder (20 minutes)
(3) Example assignment overview (10 minutes)
(4) Demonstration of problems with basic autograder (20 minutes)
(5) Strategies for better autograder design (20 minutes)
(6) Break (15 minutes)
(7) Second example assignment overview (10 minutes)
(8) Attendees develop an autograder (40 minutes)
(9) Testing each others' autograders (20 minutes)
(10) Discussion (15 minutes)

EXPECTED AUDIENCE

Anyone who teaches CS1/CS2 or introductory programming outside of undergraduate education and who is interested in either starting to use autograders or improving the quality of their autograders. We expect that this description characterizes a significant fraction of SIGCSE attendees.

PARTICIPANT EQUIPMENT REQUIREMENTS

Participants should have access to a computer with a text editor and web browser, and the ability to connect to a remote autograding server.

ADVERTISEMENT

This workshop is for instructors of introductory programming courses who are interested in adopting autograding systems or already use autograders but are dissatisfied with the quality of the feedback that they provide to students. We will demonstrate the ways in which a straightforward autograder is deficient and discuss techniques for developing autograders that are more useful to students. You will then develop your own autograder for a simple programming assignment, and test an autograder developed by another attendee. You will need a computing device with the ability to edit text documents and upload them to the cloud.

ENROLLMENT LIMIT

The default limit of 30 attendees seems reasonable.

SCHEDULING CONSTRAINTS

None.

OTHER CRITICAL INFORMATION

None.

REFERENCES

[1] Denny, P., Prather, J., Becker, B. A., Mooney, C., Homer, J., Albrecht, Z. C., and Powell, G. B. On designing programming error messages for novices: Readability and its constituent factors. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021).

[2] Douce, C., Livingstone, D., and Orwell, J. Automatic test-based assessment of programming: A review. *J. Educ. Resour. Comput. 5*, 3 (2005).

[3] D'Souza, R., Bernuy, A. Z., and Harrington, B. PyBuggy: Testing the effects of enhanced error messages on novice programmers. In *Proceedingsof the 52nd ACM Technical Symposium on Computer Science Education* (2021).

[4] Gordon, C. L., Lysecky, R., and Vahid, F. The rise of program auto-grading in introductory CS courses: A case study of zyLabs. In *2021 American Society for Engineering Education Annual Conference* (2021).