Beyond Autograding: Advances in Student Feedback Platforms

John DeNero (moderator)
University of California, Berkeley
EECS Department
Berkeley, CA 94720
+1 (415) 203-1943
denero@cs.berkeley.edu

Sumukh Sridhara
University of California, Berkeley
EECS Department
Berkeley, CA 94720
+1 (510) 642-1042
sumukh@berkeley.edu

Manuel Pérez-Quiñones
UNC Charlotte
College of Computing and Informatics
Blackburg, VA 24060
+1 (704) 687-8553
perez.quinones@uncc.edu

Aatish Nayak
Carnegie Mellon University
College of Engineering
Pittsburgh, PA 15213
+1 (848) 702 1830
aatishn@andrew.cmu.edu

Ben Leong
National University of Singapore
School of Computing
Singapore 117418
+65 6516 4240
benleong@comp.nus.edu.sg

Categories and Subject Descriptors
K.3.2 [**Computer and Information Science Education**]: Computer science education

Keywords
Computer science education, automatic grading, autograders

1. SUMMARY
Automatic grading of programming assignments has been a feature of computer science courses for almost as long as there have been computer science courses [1]. However, contemporary autograding systems in computer science courses have extended their scope far beyond performing automated assessment to include gamification [2], test coverage analysis [3], managing human-authored feedback, contest adjudication [4], secure remote code execution [5], and more. Many of these individual features have been described and evaluated in the computer science education literature, but little attention has been given to the practical benefits and challenges of using the systems that implement these features in computer science courses.

The goal of this panel is to answer common questions about how these extensions to autograding affect courses in practice. Should courses build their own submission and grading systems from scratch, or is there real benefit to using an existing platform? Are the platforms available today only applicable to specific courses? Are some features only appropriate in CS 0 and CS 1 courses? What aspects help most with scaling to large enrollments? What features are difficult to deploy or confusing to students? What are past mistakes from which the community can learn?

This panel brings together perspectives from the developers of feature-rich platforms used by multiple courses. The panelists will highlight recent research in extending autograder platforms to address related problems in CS pedagogy, as well as tips for choosing a platform and using it for the first time.

SIGCSE'17, March 8–11, 2017, Seattle, WA, USA
ACM 978-1-4503-4698-6/17/03

DOI: http://dx.doi.org/10.1145/3017680.3017686

2. SUMUKH SRIDHARA
OK – okpy.org
OK is an open source autograding system developed at UC Berkeley. It is used by the CS1 course (CS 61A) as well as four other Computer Science courses at UC Berkeley. OK supports programming projects by providing an automated tutor, a submission system, autograding, human-authored code review, and detailed analytics. As students progress through an assignment, snapshots of their in-progress work are captured and stored by the OK server along with data about passing tests and student progress. The local OK client manages project submission and exposes a variety of debugging aids for students while server-side autograding runs secret tests in sandboxed Docker containers for grading purposes.

The use of OK has made a large impact on the scalability of CS 61A to over 2,500 students per year. Debugging assistance features have reduced the amount of instructor time spent answering clarificatory questions [6].

The ability to collect in-progress work has yielded a large amount of data for instructors and researchers that was previously inaccessible when only a single final submission per student was collected. In a single offering of a 1,400 person course, OK collected over 3 million snapshots. For a single assignment, OK collected an average of 349 snapshots per student [7]. This data is used to provide instructors with reports about exactly how far along students are in the assignment, how long students are spending on each question, and common wrong answers.

Researchers have used the data from OK to integrate automatic composition feedback, targeted conceptual hints, and dynamically generated code fixes into the OK system.

Sumukh Sridhara has been a lead developer of courseware for CS 61A and the OK system at UC Berkeley, as well as head of the course teaching staff. He is currently a graduate student in the UC Berkeley EECS department.

3. MANUEL PÉREZ-QUIÑONES

Web-CAT – web-cat.org

Web-CAT's modular architecture is what makes it unique among other grading systems [8]. It can be used in courses to simply collect assignments, which can be submitted via the web or directly from various IDEs. Web-CAT can also be configured to grade assignments and provide feedback to the student. The grading can be done based on instructor provided test cases. Where Web-CAT really excels is having part of the student's grade based on how well students test their own code. Web-CAT is very flexible allowing many of its features to be configured on an assignment-by-assignment basis. It is stable and robust, and it is being used in more than 50 universities. Web-CAT is language independent; it uses a plugin architecture allowing programs in Java, C++, Scheme, Prolog and others languages to be graded automatically. As a research tool, Web-CAT has been used to explore automated feedback, semi-automated identification of bugs, gamification of feedback, and providing motivational hints.

Dr. Manuel A. Pérez-Quiñones has been affiliated with Web-CAT development since its inception. He worked closely with the lead designer of Web-CAT, Dr. Stephen Edwards, in grants, publications, and research. He used Web-CAT in many courses in the undergraduate program at Virginia Tech and has given workshops about Web-CAT at multiple universities and at SIGCSE. He is currently Associate Dean of the College of Computing and Informatics at the University of North Carolina at Charlotte.

4. AATISH NAYAK

Autolab – autolabproject.com

Autolab is an open source course management and autograding service started at Carnegie Mellon by Professor David O'Hallaron. Since its inception, it has seen over 25 contributors and is now used by the majority of computer science classes at CMU. Many courses from other schools including University of Washington, Peking University, Cornell University, among others use the service. Autolab consists of two main components: a Ruby on Rails web app, and Tango, a Python job processing server. The web app offers a full suite of course management tools including scoreboards, configurable assignments, PDF annotations, grade sheets, and plagiarism detection. The job processing server accepts job requests to run students' code along with an instructor written autograding script within a virtual machine.

Aatish Nayak serves as the project lead for Autolab. He works closely with Professor O'Hallaron and three other developers. He has used Autolab in many of his undergraduate courses and brings a student perspective to the platform's development. He is currently a 4th year undergraduate in Electrical and Computer Engineering at Carnegie Mellon University.

5. BEN LEONG

Coursemology – coursemology.org

Coursemology is an open source learning management system that incorporates gamification elements to improve student engagement. While not restricted to computer science courses, the platform has been used in a variety of CS courses, including courses on data structures, algorithms, introductory programming, and programming methodology because it includes support for the autograding of coding questions. Coursemology allows educators to add gamification elements, such as experience points, levels, achievements, to their classroom exercises and assignments. Gamification has been shown to be effective in motivating learners in curricular settings [9].

The Coursemology platform has a flexible design that does not require instructors to have a programming background. In addition to the gamification elements, Coursemology includes dashboards to monitor student progress and homework submissions, and a feed-like mechanism to allow students to communicate with instructors and clarify concepts in a convenient and timely manner.

Dr. Ben Leong is an Associate Professor of Computer Science at the School of Computing, National University of Singapore (NUS). In addition to leading development of Coursemology, he is an active member of the networking and distributed computing research community.

6. REFERENCES

[1] Jack Hollingsworth. 1960. Automatic graders for programming classes. Commun. ACM 3, 10 (October 1960), 528-529. DOI=http://dx.doi.org/10.1145/367415.367422

[2] Alexandru Iosup and Dick Epema. 2014. An experience report on using gamification in technical higher education. In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14). ACM, New York, NY, USA, 27-32. DOI: http://dx.doi.org/10.1145/2538862.2538899

[3] David Jackson and Michelle Usher. 1997. Grading student programs using ASSYST. In Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education (SIGCSE '97), James E. Miller (Ed.). ACM, New York, NY, USA, 335-339. DOI=http://dx.doi.org/10.1145/268084.268210

[4] Dejan Milojicic. 2011. Autograding in the Cloud: Interview with David O'Hallaron. IEEE Internet Computing 15, 1 (January 2011), 9-12. DOI=10.1109/MIC.2011.2 http://dx.doi.org/10.1109/MIC.2011.2

[5] Tommy MacWilliam and David J. Malan. 2013. Streamlining grading toward better feedback. In Proceedings of the 18th ACM conference on Innovation and technology in computer science education (ITiCSE '13). ACM, New York, NY, USA, 147-152. DOI=http://dx.doi.org/10.1145/2462476.2462506

[6] Soumya Basu, Albert Wu, Brian Hou, and John DeNero. 2015. Problems Before Solutions: Automated Problem Clarification at Scale. In Proceedings of the Second (2015) ACM Conference on Learning @ Scale (L@S '15). ACM, New York, NY, USA, 205-213.DOI=10.1145/2724660.2724679 http://doi.acm.org/10.1145/2724660.2724679

[7] Sumukh Sridhara, Brian Hou, Jeffrey Lu, and John DeNero. 2016. Fuzz Testing Projects in Massive Courses. In Proceedings of the Third (2016) ACM Conference on Learning @ Scale (L@S '16). ACM, New York, NY, USA, 361-367. DOI: http://dx.doi.org/10.1145/2876034.2876050

[8] Edwards, S. H., and Perez-Quinones, M. A. Web-cat: automatically grading programming assignments. In ACM SIGCSE Bulletin, vol. 40, ACM (2008), 328–328.

[9] Zachary Fitz-Walter, Dian Tjondronegoro, and Peta Wyeth. 2011. Orientation Passport: using gamification to engage university students. In Proceedings of the 23rd Australian Computer-Human Interaction Conference (OzCHI '11). ACM, New York, NY, USA, 122-125. DOI=http://dx.doi.org/10.1145/2071536.207