Online Mentorship for Computer Science Students by John DeNero, http://denero.org

Summary. We will explore the benefits of a client-server architecture for the program that students use to evaluate their progress incrementally, typically called an *autograder*. An online autograder will allow for improved feedback for students, finer-grained information for course staff, and efficient communication between the two. We hope to demonstrate that improving mentorship using this infrastructure will allow educators to engage a broad set of students interested in computer science.

Description. Programming projects are central to computer science education. In introductory courses, students typically receive incremental feedback on their progress through an *autograder* program that executes tests and identifies issues automatically. Autograders are typically offline programs, but we are interested in exploring the potential benefits of adopting an online architecture that communicates with a server each time a student executes tests.

Our goal is to improve the automated and manual mentorship of students provided by a combination of autograder feedback and interactions with course staff. Mentorship is needed to supports the broad and diverse set of students now expressing interest in computer science, many of whom have little prior computing experience. Development of an open-source implementation of these ideas is already underway[1], and we will deploy and evaluate the system during the 2014-2015 academic year for 2000 Berkeley undergraduates.

The potential benefits of an online autograder/mentorship system include:

1. **Identifying difficult topics**. By tracking and analyzing which tests cause students the most trouble, the course staff can allocate class time to remedy confusions while students are still working on their projects.
2. **Improving automated responses**. The automated feedback given to students can be updated constantly in an online architecture, for instance by linking to helpful forum posts related to a common problem shared among several students.
3. **Automated backup of student code**. By recording a server-side snapshot of students' partial solutions before each test run, the system will reduce data loss.
4. **Online code review**. Partial solutions will be published so that staff can view students' progress. We expect that if students can refer to their code in questions, and staff can refer to it in responses, instructor-student communication will be more effective.
5. **Peer instruction**. Students who complete projects before the deadline can help review and assist students who are still working, given access to their partial solutions.
6. **Discouraging guess-and-check programming.** By recording how often students run tests, we can actively discourage unproductive programming behavior, such as making arbitrary changes to an implementation until tests pass.
7. **Encouraging timely progress**. By measuring and publishing progress relative to the rest of the class, we can encourage students to start work on projects earlier by celebrating those who do start early.

[1] https://github.com/Cal-CS-61A-Staff/ok