Lessons Learned Using Automated Grading Tools to Teach Software Testing

Stephen H. Edwards
Virginia Tech
Dept. of Computer Science
edwards@cs.vt.edu
http://web-cat.org/

What is Web-CAT?

- A flexible, extendable automated grading system that can support virtually any programming language
- Completely web-based submission and feedback
- Full support for manual grading too
- All grading actions and feedback are controlled by plug-ins

Why did we develop Web-CAT?

More and more educators are adding software testing to their programming courses

- Now it's almost routine
- Tools like JUnit, and XUnit frameworks for other languages, make it much easier
- Built-in support by many mainstream and educational IDEs makes it much easier
- Many instructors have also experimented with automated grading based on such testing frameworks
- Here are our experiences in teaching test-driven development with the help of an automated grader over the past 5 years

Why have we added software testing across our programming core?

- Students cannot test their own code
- Want a culture shift in student behavior
- A single upper-division course would have little impact on practices in other classes
- So: Systematically incorporate testing practices across many courses

Practicing software testing will help students frame and carry out experiments

- The problem: too much focus on synthesis and analysis too early in teaching CS
- Need to be able to read and comprehend source code
- Envision how a change in the code will result in a change in the behavior
- Need explicit, continually reinforced practice in hypothesizing about program behavior and then experimentally verifying their hypotheses

Expect students to apply their testing skills all the time in programming assignments

- Expect students to test their own work
- Empower students by engaging them in the process of assessing their own programs
- Require students to demonstrate the correctness of their own work through testing
- Do this consistently across many courses
Test-driven development is very accessible for students

- Also called “test-first coding”
- Focuses on thorough unit testing at the level of individual methods/functions
- “Write a little test, write a little code”
- Tests come first, and describe what is expected, then followed by code, which must be revised until all tests pass
- Encourages lots of small (even tiny) iterations

Students can apply TDD in assignments and get immediate, useful benefits

- Conceptually, easy for students to understand and relate to
- Increases confidence in code
- Increases understanding of requirements
- Preempts "big bang" integration

We use Web-CAT to automatically process student submissions and check their work

- Web application written in 100% pure Java
- Deployed as a servlet
- Built on Apple’s WebObjects
- Uses a large-grained plug-in architecture internally, providing for easily extensible data model, UI, and processing features

Web-CAT’s strengths are targeted at broader use

- Security: mini-plug-ins for different authentication schemes, global user permissions, and per-course role-based permissions
- Portability: 100% pure Java servlet for Web-CAT engine
- Extensibility: Completely language-neutral, process-agnostic approach to grading, via site-wide or instructor-specific grading plug-ins
- Manual grading: HTML "web printouts" of student submissions can be directly marked up by course staff to provide feedback

Grading plug-ins are the key to process flexibility and extensibility in Web-CAT

- Processing for an assignment consists of a “tool chain” or pipeline of one or more grading plug-ins
- The instructor has complete control over which plug-ins appear in the pipeline, in what order, and with what parameters
- A simple and flexible, yet powerful way for plug-ins to communicate with Web-CAT, with each other
- We have a number of existing plug-ins for Java, C++, Scheme, Prolog, Pascal, Standard ML, ...
- Instructors can write and upload their own plug-ins
- Plug-ins can be written in any language executable on the server (we usually use Perl)

The most well-known plug-in is for grading Java assignments that include student tests

- ANT-based build of arbitrary Java projects
- PMD and Checkstyle static analysis
- ANT-based execution of student-written JUnit tests
- Carefully designed Java security policy
- Clover test coverage instrumentation
- ANT-based execution of optional instructor reference tests
- Unified HTML web printout
- Highly configurable (PMD rules, Checkstyle rules, supplemental jar files, supplemental data files, java security policy, point deductions, and lots more)
Web-CAT supports a variety of languages, and its Java plug-in is aimed at software testing:

- ANT-based build of arbitrary Java projects
- PMD and Checkstyle static analysis
- ANT-based execution of student-written JUnit tests
- Carefully designed Java security policy
- Clover test coverage instrumentation
- ANT-based execution of optional instructor reference tests
- Unified HTML web printout
- Highly configurable (PMD rules, Checkstyle rules, supplemental data files, Java security policy, point deductions, and lots more)

Web-CAT provides timely, constructive feedback on how to improve performance:

- Indicates where code can be improved
- Indicates which parts were not tested well enough
- Provides as many "revise/resubmit" cycles as possible

Assessing student tests is tricky, so we use complementary methods:

- First, we measure how many of the student's own tests pass
- Second, we instrument student code and measure code coverage while the student’s tests are running
- Third, we use instructor-provided reference tests to cross-check the student’s tests
- We multiply the percentages together, so students must excel at all three to increase their score

Students improve their code quality when using Web-CAT:

Let’s see it live!

- Time for a demo …

Some lessons learned …

- We've had many successes … and most failures come from lack of testing knowledge among instructors!
- Tougher coverage requirements seem to help, even for beginning students
  - … but you need to deal with situations where students may not be expected to cover some code.
- Preventing students from cheating the system
- Static analysis checks work well for this
**Some lessons learned ...**

- Typically need reference tests in addition to student-written tests
  - Or a reference implementation, depending on the language.
- Can't give away all reference test results, or students won't write their own tests
- ... But students need some behavioral feedback to reduce frustration and provide direction

**Conclusion: including software testing helps promote learning and performance**

- If you require students to write their own tests ...
  - Our experience indicates students are more likely to complete assignments on time, produce one third less bugs, and achieve higher grades on assignments
  - It is definitely more work for the instructor
  - But it definitely improves the quality of programming assignment writeups and student submissions

**Visit our SourceForge project!**

- http://web-cat.org/
  - Info about using our automated grader, getting trial accounts, etc.
  - Movies of making submissions, setting up assignments, and more
  - Custom Eclipse plug-ins for C++-style TDD
  - Links to our own Eclipse feature site and our SourceForge downloads