Automatically Grading Programming Assignments with Web-CAT

http://web-cat.org/cta13

Stephen Edwards

Dept. of Computer Science

Virginia Tech

What is Web-CAT?

- A plug-in-based web application
- Supports **electronic submission** and **automated grading** of programming assignments
- Fully customizable, scriptable grading actions and feedback generation
- Lots of support for grading students based on **how well they test their own code**

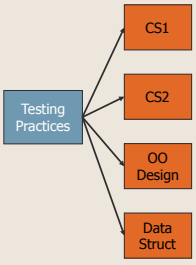
Who uses Web-CAT?

- About **80 institutions** and growing
- **14,448 users** on our servers, approaching **20K users** worldwide
- Since 2003, Virginia Tech's servers alone have processed approximately:
  - **964,926** program submissions
  - By **14,448** users
  - In **562** course sections

More educators are adding software testing to their program-ming 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 **my experiences** in teaching test-driven development with the help of an automated grader over the past 10 years

Why have we added software testing across our programming core?



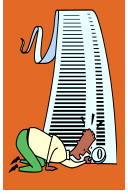
- Students **cannot test** their own code
- Want a **culture shift** in student behavior
- A single upper-division course would have **little impact** on practices in other classes
- So: Systematically incorporate testing practices across many courses

Software testing helps 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 **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**

Expect students to apply testing skills all the time



What kinds of assign-ments?

- Regular CS1 and CS2 assignments (of course!)
- Text adventure games
- Greenfoot-style micro-worlds
- **Asteroids**, MineSweeper
- AI computer players for Battleship!, Tetris, and more
- Random maze explorers
- Swing **GUI applications** (even 2D drawing editors)
- **Android apps** (even 2D and physics-based games, and map-based geotagged photo apps)
- Parsers and interpreters for PL courses

What tools and tech-niques should we teach?

- We want to start with skills that are **directly applicable** to authentic student-oriented tasks
- Don't want to add bureaucratic **busywork** to assignments
- Without tool support, this is a lost cause!
- It is imperative to give students skills they **value**
- … But most textbooks only give a "conceptual" intro to idealized industrial practices, not techniques students can use in their own assignments

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

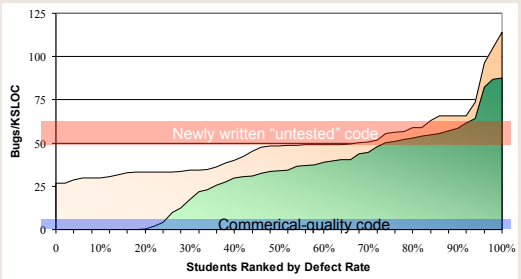
Students can apply TDD and get immediate, useful benefits

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

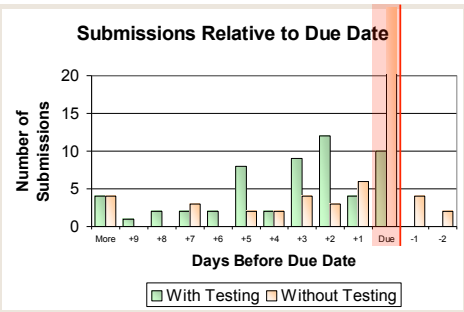
TDD tools are widely, freely available

- Lots of open-source tools, particularly for OO languages
- JUnit (for Java): http://junit.org/
- XUnit links (for other languages): http://xprogramming.com/software/
- We use tools like this for Java, C++, Scheme, Prolog, Haskell, and even Pascal in our courses

Students improve their code quality when required to test



Students start earlier and finish earlier



We use Web-CAT to automatically check student 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

Grading plug-ins are the key to Web-CAT's flexibility and extensibility

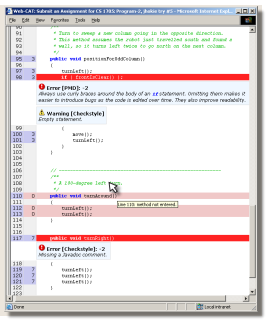
- Processing for an assignment consists of a **"tool chain"** or **pipeline** of one or more grading plug-ins
- The instructor has complete control over which plug-ins appear in the pipeline, in what order, and with what parameters
- A simple and flexible, yet powerful way for plug-ins to communicate 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)

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

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

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



- A **course** is an academic course that can be offered over and over
- A **course offering** is a specific offering of a course during a specific semester or term
- An **assignment** is a reusable set of instructions and grading procedures/criteria
- An **assignment offering** is a specific offering of an assignment within a specific *course offering* (with a **due date**)

First, some basic Web-CAT terms

For Java, Web-CAT provides three main features you can combine for grading

- The **instructor** can write reference tests

... **or not**

- The **student** can write his/her own software tests

... **or not**

- **Static analysis tools** can check coding style

... **or not**

Let's see it working!

- All of today's examples are on the web:

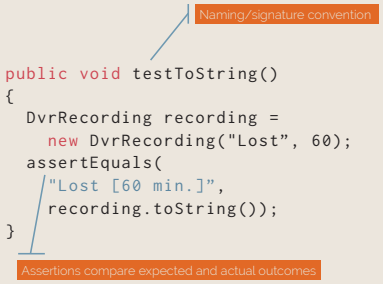
http://web-cat.org/cta13

Suppose we have a class for DVR record-ings

```
public class DvrRecording
{
  private String title;
  private int duration;

  public DvrRecording(
    String title, int duration)
  {
    ...
  }

  public String getTitle() { ... }
  public int getDuration() { ... }
  public String toString() { ... }
}
```

```
public void testToString()
{
  // 1. Initial conditions
  DvrRecording recording =
    new DvrRecording("Lost", 60);

  // 2. Action to test
  String output =
    recording.toString();

  // 3. Check expected results
  assertEquals(
    "Lost [60 min.]", output);
}
```

A test might look like this



The same, but shorter

```
private DvrRecording recording;

// Initial conditions for all tests
public void setUp()
{
  recording =
    new DvrRecording("Lost", 60);
}

public void testToString()
{
  assertEquals(
    "Lost [60 min.]",
    recording.toString());
}
```

Always starts in a clean starting state

With common setup factored out

```
private DvrRecording recording;

@Before
public void setUp()
{
  recording =
    new DvrRecording("Lost", 60);
}
@Test
public void testToString()
{
  assertEquals(
    "Lost [60 min.]",
    recording.toString());
}
```

Annotations instead of inheritance

No more naming conventions

The same, but in JUnit 4

Let's see it working!

- All of today's examples are on the web:

  http://web-cat.org/cta13

Walkthrough wrap-up

- Time for questions about the steps we have demonstrated …
- … or questions about how to use it with your own assignments

The most important step in writing testable assignments is …

- Learning to write tests yourself
- Writing an instructor's solution **with tests** that thoroughly cover all the expected behavior
- Practice what you are teaching/preaching
- Extra effort before assignment is "opened" (more prep time) but less effort after assignment is due (less grading time)

How do you write tests for:

- Exceptional conditions
- Main programs
- Code that reads/write to/from stdin/stdout or files
- Assignments with lots of design freedom
- Code with graphical output
- Code with a graphical user interface

Areas to look out for

Our testing library provides ...

In our student.jar library:

- Set **stdin** in test cases
- Get history of **stdout** (cleanly reset for each test)
- Newline normalization for output
- System.**exit()** throws exception
- Better error messages for student assertion mistakes
- "**Fuzzy**" string matching (ignore caps, punctuation, spacing, etc.)
- **Regular expression** and fragment matching
- Adaptive **infinite loop** protection during grading
- Swing GUI testing through **LIFT**

Lessons learned writing testable assignments

- Requires greater clarity and specificity
- Requires you to explicitly decide what you wish to test, and what you wish to leave open to student interpretation
- Requires you to unambiguously specify the behaviors you intend to test
- Requires preparing a reference solution before the project is due, more upfront work for professors or TAs
- Grading is much easier as many things are taken care by Web-CAT; course staff can focus on assessing design

If you give students tests instead of writing their own

- Students appreciate the feedback from tests, but will **avoid thinking** more deeply about the problem
- Seeing the results from a complete set of tests discourages student from thinking about how to check about their solution on their own
- This **limits the learning benefits**, which come in large part from students **writing their own** tests
- Lesson: balance providing suggestive feedback without "giving away" the answers: **lead the student** to think about the problem

Conclusion: including software testing promotes 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

It is time for any final questions ...

- About anything covered ...
- About how I've used these techniques in courses
- About how we start our freshmen out in the very first lab
- About the availability of Web-CAT
- ... Or anything else you want to ask

Thank You!

- Our community is our most valuable asset!

http://web-cat.org