Crossing the Software Education Chasm Using SaaS & Cloud Computing

Armando Fox & Dave Patterson, {fox,patterson}@cs.berkeley.edu

HPTS “poster” session

Graphic: Google’s “Testing on the Toilet” program
What’s the problem?

• New dev hires good at coding, debugging*

• **New hires are NOT** good at:
  – testing
  – working in teams
  – communicating with customers
  – dealing with legacy software

• Complaint: SWE courses teach outdated methodologies, avoid these real problems

• Constraint: Undergrad spends ~12 hr./week per course = 3 work-weeks during semester

*Sources: Google, eBay, Amazon, Microsoft, Facebook, & Begel et al. 2008 study
Approach: SW Eng. With SaaS

- Teach SaaS & cloud computing using Rails
  - **High productivity**
  - Great testing tools, ease of public deployment
- Emphasizes testing at all levels
  - 2 of 16 projects: 100% unit test (statement) coverage, 100's integration tests
- Small-team, Agile development
  - Projects varied in amount of functionality—not code quality
  - Agile is great fit for classroom
A typical 1-week iteration (6-8 times in 1 semester course)

Feature idea

Lo-fi UI sketches, storyboards

XHTML & Cascading Stylesheets

User stories & scenarios

Scenarios => Class-Responsibility-Collaborator Cards

Scenarios => Design patterns

Behavior Driven Design: identify code you wish you had

Test Driven Development to get code running

Compute Velocity (rate of feature delivery)

Deploy using cloud computing
Learn By Doing

- Software arch., design patterns, coding practices (10 hr.)
- Test-first development, unit testing (5 hr.)
- Behavior-driven design, integration testing (3 hr.)
- Agile, iteration-based project management (2 hr.)
- Version management & collaboration skills (3 hr.)
- SaaS technologies, deployment & operations (7 hr)

- Ruby & Rails
- RSpec
- Cucumber
- Pivotal Tracker
- Git & Github
- Cloud computing: EC2, Heroku
Feature: staff can add admit to meeting with open slot
As an EECS staff member
So that I can accommodate last-minute requests
I want to manually tweak a faculty member's schedule

Scenario: add an admit to a meeting with an open slot
Given "Velvel Kahan" is available at 10:20
When I select "Velvel Kahan" from the menu for the 10:20 meeting with "Armando Fox"
And I press "Save Changes"
Then I should be on the master meetings page
And I should see "Velvel Kahan added to 10:20AM meeting."
And "Armando Fox" should have a meeting with "Velvel Kahan" at 10:20

Scenario: remove admit from meeting

...etc.
Survey Results from 2 sets of students (Fall 09, Fall 10)
What was useful?

<table>
<thead>
<tr>
<th>Topic</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version control</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Working in a team</td>
<td>93%</td>
<td>7%</td>
<td>0%</td>
</tr>
<tr>
<td>SaaS knowledge (used in other languages/frameworks)</td>
<td>87%</td>
<td>7%</td>
<td>7%</td>
</tr>
<tr>
<td>Design patterns</td>
<td>80%</td>
<td>13%</td>
<td>7%</td>
</tr>
<tr>
<td>Unit testing skills (mocking, stubbing, test automation, ...)</td>
<td>75%</td>
<td>19%</td>
<td>6%</td>
</tr>
<tr>
<td>Ruby on Rails</td>
<td>73%</td>
<td>20%</td>
<td>7%</td>
</tr>
<tr>
<td>Cloud computing knowledge</td>
<td>67%</td>
<td>33%</td>
<td>0%</td>
</tr>
<tr>
<td>Test-first development</td>
<td>67%</td>
<td>20%</td>
<td>13%</td>
</tr>
<tr>
<td>User stories</td>
<td>60%</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>Lo-fi UI mockups</td>
<td>44%</td>
<td>25%</td>
<td>31%</td>
</tr>
<tr>
<td>Velocity</td>
<td>21%</td>
<td>57%</td>
<td>21%</td>
</tr>
</tbody>
</table>
Legacy Code

- All industrial partners concerned about this
- Proposal: use Feathers’s approach in *Working Effectively with Legacy Code*
  - “Legacy code == code without tests”
- Have students add feature to large (~30 KLOC) existing code base

<table>
<thead>
<tr>
<th></th>
<th>Good code</th>
<th>Not-so-good code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has tests</td>
<td>Use TDD</td>
<td>Refactor first, then TDD</td>
</tr>
<tr>
<td>Lacks tests</td>
<td>Create tests, then TDD</td>
<td>Create tests so can refactor, then TDD</td>
</tr>
</tbody>
</table>
I’m writing more lines of tests than lines of code
We're not writing much code, just reusing a lot of stuff  
(~50% of projects integrate with external API like Facebook, Google Maps, etc.)
BDD is weird, usually I just sit down and start coding
What do you mean, we have to search for answers on  
[Google, StackOverflow, PeepCode, blogs, ...]
I don’t grok language feature X
  (trans.: X isn’t in Java or C++)
Learning curves for Ruby, Rails, tools
  (learning on job, multi-language houses the norm)
The SaaS Love Triangle

High productivity, cloud deployment ⇔ Projects work & have immediate, global visibility

“Frequent release” culture ⇔ weekly progress during semester

“Incremental” testing discipline & great testing tools ⇔ testing as root of process, not after-chore