2012 Grace Hopper Celebration of Women in Computing

Everything I needed to know about software testing I learned in … wait, nobody taught me about software testing!

Fiona Tay[1] and David Janzen[2]
[1] Fiona Tay, Pivotal Labs, San Francisco, CA 94103
[2] California Polytechnic State University, San Luis Obispo, CA 93407

PANEL OBJECTIVE:
Our goal in this panel is to communicate the importance of structured software testing for software quality and investigate how this crucial skill is portrayed and taught in undergraduate CS curricula. We will present different approaches to teaching testing in an undergraduate computer science/software engineering curriculum. We will share our experiences as software engineers who write tests, and how our college experience did or did not develop these skills.

TARGET AUDIENCE:
Our target audience ranges from faculty who teach software engineering, engineers who test software, professionals interested in software quality, and students wondering what they need to know about testing and why.

PANEL FORMAT:
This will be a 60-min panel comprised of academics and software engineers from different universities and organizations. The time will be evenly split between remarks from panelists and discussion. First, each panelist will briefly speak for approximately five minutes on the panel topic specific to their role as faculty or industry practitioner, and reflecting on their experience with testing as an undergraduate student. Then, the panel will take questions centered around the three topics of testing's place in a CS curriculum, innovations in testing pedagogy, and student exposure to testing in industry. The audience will be invited to ask questions and share ideas.

OVERVIEW:
The role of software testing in ensuring software quality is well-established (McConnell, 1993). Testing is an essential step in the software development life-cycle, and tests have been recommended for various contexts - new web applications to large legacy systems - and for various purposes - isolating bugs to documenting code. However, in practice, software is not tested consistently. Informal polls show that up to 40% of software projects are completely untested (Hacker News, 2012).

Additionally, despite its critical importance in industry, software testing remains a small component of most CS curricula. One common approach is to integrate testing into a software engineering course, and this has met with mixed success. At Harvey Mudd College, students in the required software engineering course must devise a testing plan and implement it while developing features. In practice, students tend to write few tests and regard testing as a nuisance. In general, students often fail to appreciate the value of tests perhaps because they rarely have to live with software beyond the end of an academic term.

Alternative approaches that holistically incorporate tests throughout CS curriculum have been suggested (Janzen, 2006) and occasionally implemented. As agile methodologies become more popular in industry, software testing is receiving renewed interest among faculty. Incorporating testing into courses is supported by automated tools (Edwards, 2004), which have emerged and matured in recent years. These tools have enabled the addition of testing tasks into large introductory classes by automating the assessment of test coverage. In trials, intrinsic approaches have improved the quality of

student submissions without sacrificing course content or instructional time. (Janzen, 2006)

Unfortunately, gaining buy-in from a wide range of faculty who often have little professional software experience and poor software testing training can be difficult. Consequently, these approaches have not been widely adapted.

In addition, little progress has been made towards conveying the role of testing in large software projects. In reality, developers lack the resources to reach 100% test coverage and the process of planning testing requires judgment calls and trade-offs. Product managers must decide which parts of the software to test and how to test them. This aspect is rarely addressed in software engineering courses, whose projects are often too small to illustrate this principle.

All our panelists have experience with writing tests and with computer science education, be it as a student or as a professor. They have found ways to develop proficiency in testing either through curricular innovations, industry experience, or applying skills from other domains.

PANEL QUESTIONS:

After remarks from individual panelists, the audience will be invited to share opinions and pose questions related to each of these three domains. Examples of possible questions follow.

<u>On Developing Future Testers</u>

- Who should learn testing? Should all students learn testing/have to write tests?
- Are our current testing pedagogies serving students well?
- How do engineers become interested in testing outside of school? How do they learn good testing strategies?
    - Follow up: What courses outside of CS could help students become better testers? What courses *inside* CS should future testers take to strengthen their skills? E.g. user interfaces, software usability.
    - Follow up: is broader better? Can students leverage a broader education/range of non-academic experiences to improve their ‘test sense’?

<u>On Developing Better Test Curriculum</u>

- What challenges do instructors face in teaching testing? How can we overcome them?
    - Tests are not incentivized: Make them a substantial component of course grade
    - Students don’t know how to write tests: Language choice, explicit instruction
    - Grading tests is resource-intensive: Automatic test grading
        - What tools exist that support testing in courses? (e.g. Web-CAT)
        - How might Massive Open Online Courses (MOOCs) affect the importance of testing in courses?
- In software engineering courses, students are told of the value of testing, but this is in the abstract. How can we help students understand the real, tangible value of testing?
    - Is this problem inherent in the choice of greenfield projects?

<u>On Bringing Students into Industry as New Testers</u>

- Few students even consider testing as a career, and they often see testing as a second-class of opportunity compared to development. How can we convince more students to explore test roles in industry?
- What can industry mentors (intern coaches, managers, leads) do to help develop students’ test skills while they are interning or doing co-ops?

2012 Grace Hopper Celebration of Women in Computing

PANELISTS:

Ruth Doane, Google (ruthdoane@google.com)

Ruth Doane has been a software engineer at Google for four years, and now works on the Analytics team, in Irvine, CA. She has also worked on custom software for the design of high-end microprocessors at Motorola, and developed online games at a small studio that she and her husband started in Austin. Some time in the previous millennium, she studied math and computer science at Carnegie Mellon University. At Google, she enjoys an engineering culture that resembles a start up combined with large scale data challenges.

David Janzen, California Polytechnic State University: San Luis Obispo (djanzen@calpoly.edu)

David Janzen is a computer science professor at Cal Poly SLO where he incorporates test-driven development (TDD) and automated testing into almost every course he teaches. His PhD examined the effects of TDD on software design quality, and he has numerous publications on TDD in education. David has worked as a software developer and manager at Sprint, and has moonlighted for over a decade as a consultant and professional trainer. Most recently David has been busy teaching Android courses and creating WebIDE, an NSF-sponsored system for hosting interactive student labs that support a test-driven learning approach.

Ciera Jaspan, California Polytechnic State University: Pomona (ciera@csupomona.edu)

Ciera Jaspan is an assistant professor of software engineering at Cal Poly Pomona. She has taught software engineering classes at Carnegie Mellon and Cal Poly Pomona. Her PhD thesis was on the static analysis of software frameworks to detect misuses of complex protocols. She has also designed software engineering tools for eBay to integrate program correctness checks into a tight development life-cycle.

Fiona Tay, Pivotal Labs (me@fionatay.com)

Fiona Tay is an agile software engineer at Pivotal Labs, where she develops Ruby on Rails applications. She first wrote tests during a summer internship at Pivotal Labs, which convinced her of their utility. As chief bug-fixer in a summer REU, she introduced a test framework into a six-year-old tablet application, which was used to isolate bugs and bring the software to release. She is also a founding member of the Claremont Women in Computing group.

Kate Tsoukalas, Microsoft (ktsouka@microsoft.com)

Kate Tsoukalas is a Software Development Engineer in Test at Microsoft working on Windows Phone. She holds a Master of Computing Science degree from Simon Fraser University; her key interests include usability and interaction design, testing techniques and methodologies, and anything to do with mobile software development. She is also passionate about encouraging, promoting, and increasing the number of women in STEM fields, particularly Computing Science.

REFERENCES:

Anonymous. "Hacker News Poll: Do You Test Your Code?" Hacker News. Hacker News. Web. 15 Mar. 2012. <http://news.ycombinator.com/item?id=3702827>.

Edwards, Stephen H. "Using Software Testing to Move Students from Trial-and-error to Reflection-in-action." ACM SIGCSE Bulletin 36.1 (2004): 26. Print.

Janzen, David S., and Hossein Saiedian. "Test-driven Learning." ACM SIGCSE Bulletin 38.1 (2006): 254. Print.

McConnell, Steve. Code Complete: A Practical Handbook of Software Construction. Redmond, WA: Microsoft, 1993. Print.