Panel: Teaching Software Testing from an Industry Perspective

Dr. Mark C. Paulk

CSEE&T 2016, Dallas

Teaching Undergraduates Software Testing Techniques

Teaching from the textbook

- **black box –**
  - **requirements coverage, equivalence class partitioning, boundary value analysis, predicate-based testing (BOR, BRO, BRE, MI)**
- **white box**
  - **statement coverage, decision coverage, condition coverage, MC/DC, c-use, p-use, all-uses, mutation testing**

Supplemental information, e.g., peer reviews, technical debt, test-driven development, …

Teaching What's Needed?

To what degree are the various techniques primarily of academic interest?

Should an undergraduate course focus on the pragmatic needs of industry?

Should the "advanced topics" be the focus of the graduate courses?

Shouldn't advanced topics move into industrial use if they truly add value?

Questions for Industry

What testing techniques are used in your software projects?

What testing techniques do you find your new hires inadequately prepared to use?

What testing tools have you found useful in doing real-world testing?

What testing needs do you have that your current staff are not well-prepared to address – perhaps needing further research (and tool development) to adequately address?

Questions for Academia

What testing techniques do you think should be taught to undergraduates?

Graduates?

What testing techniques are you researching that you think are ready for industry piloting?

What testing techniques do you think should be more broadly used in industry?

The Bottom Line

Are we teaching our students what they need to know about testing?

Are there any "popular" testing techniques that are of little practical value?

Are there any infrequently used testing techniques that should be strongly advocated for industry use?

Panelists

Mark Paulk, UT Dallas (moderator)

Anthony Adesanwo, Match Group

Vidroha Debroy, NeoTek Energy

Dennis Frailey, retired Raytheon Fellow

Laura Henning, Parker Hannifin Aerospace

Tom Wissink, retired Lockheed Martin

Eric Wong, UT Dallas

Teaching Software Testing

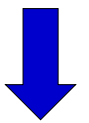
Dennis J. Frailey
(Retired) Principal Fellow, Raytheon Company
Adjunct Professor of Computer Science and Engineering, SMU
Adjunct Professor of Computer Science and Engineering, UT Arlington
Frailey@ACM.ORG
Frailey@Lyle.smu.edu
Dennis.Frailey@UTA.EDU

Presented at CSEET 2016

What Testing Techniques Are Used in Your Software Projects?

Most of our projects involve development of new hardware as well as software

Most require high reliability and some are safety critical



- ***System level requirements*** **are just as important as software requirements**
- **We often require use of *special purpose testing equipment* that is unique to the product being developed**
- **Test *procedures and results must often be carefully recorded (documented)* and are often audited**

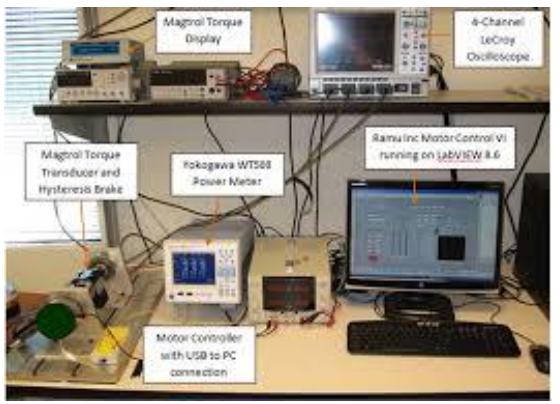
What Testing Techniques Do New Hires Not Know?

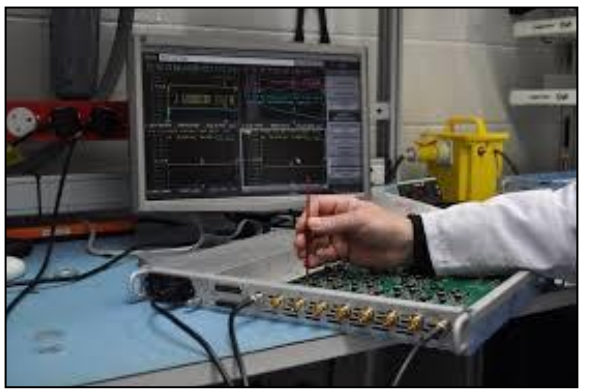
Results vary, but:

- **Many know software but know *too little about hardware***
- **Some don't seem to understand the *underlying principles* – only how to use specific languages or tools**
- **Few of them know *testing discipline*:**
  - **Developing effective test plans**
  - **Documenting the results of testing (test reports)**
  - **Testing the test code**

What Testing Tools Do You Use?

- **Product-specific Test Equipment and Tools**
  - **Software staff must participate in design and definition of these tools**





What Testing Tools Do You Use?

- **Product-specific Test Equipment and Tools**
  - **Software staff must participate in design and definition of these tools**

- **Simulation**
  - **Simulate the system before building it**
  - **Compare actual results with simulated results to find underlying causes of problems**

What Testing Tools Do You Use?

- **Product-specific Test Equipment and Tools**
  - **Software staff must participate in design and definition of these tools**
- **Simulation**
  - **Simulate the system before building it**
  - **Compare actual results with simulated results to find underlying causes of problems**
- **Automated Testing / Test Generation**
  - **Requires very good discipline in defining requirements**
  - **Can potentially save a lot of time and money**

What Are Your Testing Needs Where Current Staff are Not Well Prepared and Research is Needed?

Integration of hardware and software testing methods and tools

- **The problems are often complex and may involve both hardware and software failures**

Automatic testing

- **Sometimes the system is too high in performance to do traditional testing**
- **Examples:**
  - **With high speed networking, you may need to see and track individual packets**
  - **With multi-core systems the interactions between cores can be very hard to see with traditional testing methods**

Bottom Line

You Must Consider the Complete System/Product, Not Just the Software

- **Software developers must understand how the hardware works**

Testing Begins At the Start of the Project

- **Test plans should be initiated by system/software designers, not by "testers".**
- **"Test First" – don't write code if you don't know how you will test it**
- **Requirements may change, but you must make them as specific as possible**
  - **Are the requirements testable?**
  - **Are they unambiguous?**

Questions?