Teaching the Software Testing Course: A Tutorial

Cem Kaner, J.D., Ph.D.
Florida Institute of Technology
kaner@kaner.com

Abstract

This tutorial provides materials and supporting discussion for teaching the software testing course. Readers who see this description after CSEE&T 2004 is complete, can find the latest set of course materials at www.testingeducation.org.

1. Introduction

This tutorial is intended to help you think through the attributes of the software testing course(s) that you might teach or support at your university.

I teach two primary courses in testing (both required for Florida Tech's B.Sc. in Software engineering). The first studies black box testing, the second a mix of glass box testing and test-driven programming. We offer several other test-related courses focusing on security-related testing, mathematics underlying certain testing models, and special topics of interest in the current year. This gives me access to a variety of support material. Lecture notes donated by other teachers (at www.testingeducation.org) supplement that further.

My foundation vision is that testers are technical investigators, who use a wide array of techniques to discover quality-related information about a product under test. I stress critical thinking, risk analysis, exploratory bug-hunting, and persuasive communication over routine procedure.

The tutorial considers the course from five angles:

- Course scope and design
- Assignments
- Exams
- Available materials

2. Course scope and design

If I was teaching only one testing course, it would be black box, evaluating the program from the outside. I'd rather focus students on what's actually done in industry—and how to do it very well—than on programming or on interesting, but primarily theoretical, issues.

Other instructors prefer glass box techniques or hope to achieve a blend.

My goal in this section is a discussion. I'll explain my preference and my concerns about alternate approaches. I hope other attendees will share their experiences and scope reasoning.

3. Assignments

I prefer *authentic performances*, in Wiggins' [10] sense—realistic, meaningful tasks. We pick an open source product under development and apply an ongoing series of test techniques to it. We report bugs in their bug tracking database and get feedback from their

programmers. In the glass box class, we take on a much smaller program, probably an open source test tool, reverse engineer it, enhance it, evaluate and enhance its unit tests and perhaps its API-level tests.

We'll share experiences with these and other types of assignments, and consider the logistics (facilities, hardware, etc.) needed to support this work.

4. Exams

I hand out a study guide near the start of the term. The guide has many questions, including definitions, short answer essays or derivations, and longer answer essays, graphs, or derivations. The exam questions are drawn from the study guide pool. This encourages students to study everything I want them to study in the course, to prepare carefully thought out answers, and to collaborate with other students. I can and do demand and get high quality answers on exams, but it takes a lot of coaching to help students learn how to provide those answers.

5. Available materials

Attendees will receive a CD with 750 slides of lecture notes, a few dozen supporting papers, and pointers to other papers useful to assign as required or supplementary reading. There's more material here than you can cover in a term, but what's here may cover many or most of the topics you want in your course.

6. Acknowledgements

The black box testing course has matured over a 10-year period, with the help of many commercial and academic colleagues. I've taught variations of the course over 100 times as professional development short courses (8 to 40 lecture hours), frequently co-teaching with Hung Quoc Nguyen [7, 8, 9], Doug Hoffman [3], James Bach [1, 5, 6] and Elisabeth Hendrickson [2]. IBM/Rational's course, *Principles of Software Testing for Testers* [4], is a customized version of this course. I've also taught an academic version of the course seven times, supervised the teaching of it by Pat McGee, and am now helping other instructors adapt the course to their schools. Recent course development has been partially supported by NSF Grant EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers."

7. References

[1] Bach, J. website, http://www.satisfice.com
[2] Hendrickson, E. http://www.qualitytree.com/
[3] Hoffman, D. http://softwarequalitymethods.com/
[4] Kaner, C. Principles of Software Testing for Testers, course available from IBM/Rational, http://www.rational.com/university/paths/tester.jsp
[5] Kaner, C. & Bach, J. Black Box Testing: Commercial Course Notes, 2003, http://www.testingeducation.org/coursenotes
[6] Kaner, C., Bach, J. & Pettichord, B. Lessons Learned in Software Testing, John Wiley & Sons, 2001
[7] Kaner, C., Falk, J., Nguyen, H.Q. Testing Computer Software, 2d ed., Van Nostrand Reinhold 1993, reprinted John Wiley & Sons, 1999
[8] Nguyen, H.Q., www.logigear.com
[9] Nguyen, H.Q., Johnson, B. & Hackett, M. Testing Applications on the Web: Test Planning for Mobile and Internet-Based Systems, Second Edition, 2003
[10] Wiggins, G. Educative Assessment, Jossey-Bass, 1998