

# Black Box Software Testing

Fall 2005

## Overview for Teachers

Cem Kaner, J.D., Ph.D.  
Professor of Software Engineering  
Florida Institute of Technology  
and  
James Bach  
Principal, Satisfice Inc.

### **Copyright (c) Cem Kaner & James Bach, 2000-2005**

This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

These notes are partially based on research that was supported by NSF Grant EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

# About Cem Kaner

[www.kaner.com](http://www.kaner.com)

My current job titles are Professor of Software Engineering at Florida Institute of Technology, and Research Fellow at Satisfice, Inc. The theme of my career is satisfaction and safety of software customers and workers.

I've worked as a programmer, tester, writer, user interface designer, software salesperson, organization development consultant, teacher, as a manager of user documentation, software testing, and software development, and as an attorney focusing on the law of software quality. These have provided many insights into relationships between computers, software, developers, and customers.

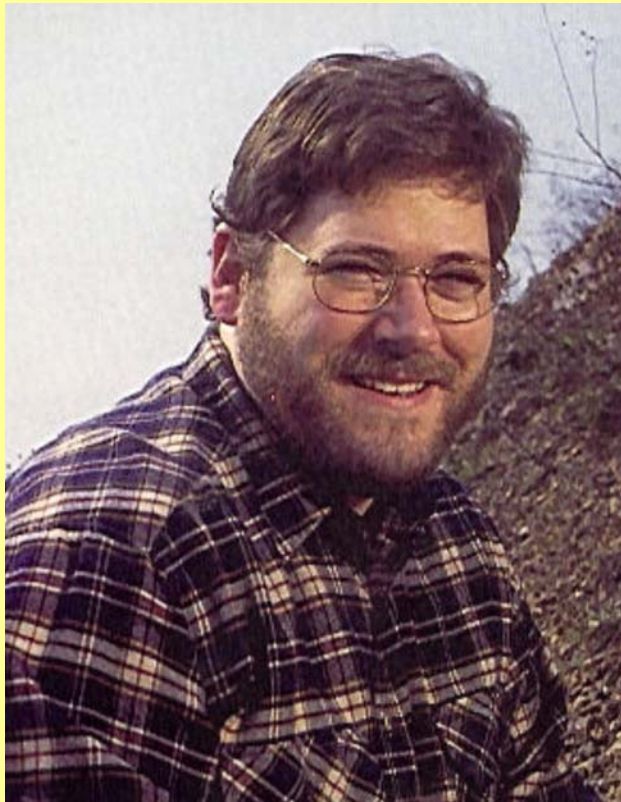
I'm the senior author of three books:

- ***Lessons Learned in Software Testing*** (with James Bach & Bret Pettichord)
- ***Bad Software*** (with David Pels)
- ***Testing Computer Software*** (with Jack Falk & Hung Nguyen).

I studied Experimental Psychology for my Ph.D., with a dissertation on Psychophysics (essentially perceptual measurement). This field nurtured my interest in human factors (and thus the usability of computer systems) and in measurement theory (and thus, the development of valid software metrics.)

# About James Bach

[www.satisfice.com](http://www.satisfice.com)



I started in this business as a programmer. I like programming. But I find the problems of software quality analysis and improvement more interesting than those of software production. For me, there's something very compelling about the question "How do I know my work is good?" Indeed, how do I know anything is good? What does good mean? That's why I got into SQA, in 1987.

Today, I work with project teams and individual engineers to help them plan SQA, change control, and testing processes that allow them to understand and control the risks of product failure. I also assist in product risk analysis, test design, and in the design and implementation of computer-supported testing. Most of my experience is with market-driven Silicon Valley software companies like Apple Computer and Borland, so the techniques I've gathered and developed are designed for use under conditions of compressed schedules, high rates of change, component-based technology, and poor specification.

# Credits & Legal Notices

These notes were originally developed in co-authorship with Hung Quoc Nguyen. James Bach has contributed substantial material. I also thank Jack Falk, Elizabeth Hendrickson, Doug Hoffman, Bob Johnson, Brian Lawrence, Pat McGee, Melora Svoboda, and the participants in the Los Altos Workshops on Software Testing and the Software Test Managers' Roundtables. Additional acknowledgements appear on specific slides.

These notes include some legal information, but you are not my legal client. I do not provide legal advice in the notes or in the course. Even if you ask me a question about a specific situation, you must understand that you cannot give me enough information in a classroom setting for me to respond with a competent legal opinion. I may use your question as a teaching tool, and answer it in a way that I believe would “normally” be true but my answer may be inappropriate for your particular situation. I cannot accept any responsibility for any actions that you might take in response to my comments in this course. If you need legal advice, please consult your own attorney.

The practices recommended and discussed in this course are useful for an introduction to testing, but more experienced testers will adopt additional practices. I am writing this course with the mass-market software development industry in mind. Mission-critical and life-critical software development efforts involve specific and rigorous procedures that are not described in this course.

# The Creative Commons License

“This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.”


- You are welcome to use these slides. You don't need our permission. You **do** have to include our copyright notice (give us credit for our work).
- You are welcome to modify the slides. However, if you distribute the **modified** slides, you must distribute them under this same Creative Commons license.
- If you need to distribute a modified set of slides under your own trade name or under a restrictive license, that might be possible. Contact us to ask for permission.

# Center for Software Testing Education & Research

Our mission is to create effective, grounded, timely materials to support the teaching and self-study of software testing, software reliability, and quality-related software metrics.

Florida Tech's Center for Software Testing Education & Research formed in November 2003, as a collaboration among Cem Kaner, Ph.D., J.D. (Professor) (Director), Walter P. Bond, Ph.D. (Associate Professor), Scott Tilley, Ph.D. (Associate Professor), Michael Andrews, Ph.D. (Assistant Professor), James Whittaker, Ph.D. (Professor)

We also collaborate closely with James Bach (Satisfice), Bret Pettichord (Thoughtworks), Douglas Hoffman (Software Quality Methods) and Pat Schroeder, Ph.D. (Milwaukee College of Engineering: Computer Science)



**Donations to the  
Center are most  
welcome. Please  
contact us at  
[kaner@cs.fit.edu](mailto:kaner@cs.fit.edu).**

# Our learning objectives

Testing software involves investigating a product under tight constraints. Our goal is to help you become a better investigator:

- Develop skills with several techniques
- Choose effective techniques for a given objective under your constraints
- Improve the critical thinking and rapid learning skills that underlie good testing
- Communicate your findings effectively
- Plan investments (in documentation, tools, and process improvement) to meet your actual needs
- Create work products that you can use in job interviews to demonstrate testing skill

**As a teacher, you have your own learning objectives. From this course, pick what advances those objectives and skip the rest. You can also modify our slides, create your own slides, and create your own videos and exercises.**

# Our teaching approach

## Spread the lessons over time

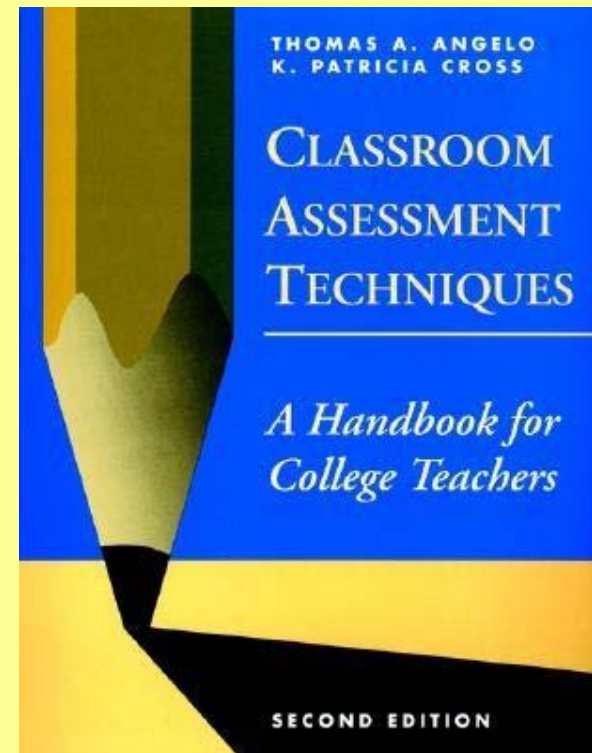
- 3-day classes are necessary evils for commercial courses because of the costs of travel and consultant time, but they cram too much information into too little time, with too little opportunity for application, practice, or debate.
- The web-based course was *designed* as an alternative. Try one topic a week or even one topic per month. In the interim, find ways to apply the information, to evaluate it, to discuss it. This is how to learn technical material. If a topic isn't worth the time needed to learn it, it isn't worth the lecture time. Skip it, do a different topic instead.

# Our teaching approach

## Active participation is very important

- Quizzes and assignments and tests are teaching tools, not just evaluation tools.
  - Students often think they “know” something long before they can do it (before they have developed the skill) and before they can explain it themselves or decide intelligently when and why to use it.
  - Tests and tasks help the student apply the ideas, focus on interesting questions and confront interesting issues.
  - Choose your questions and assignments with care.

Evaluations don't have to carry grades or have pass/fail implications. Especially in a commercial course, evals are for feedback, not for grading.



# Our teaching approach

## Give feedback

- Look at your students' work, tell them what you see. Feedback is a foundation for improvement.
- Have students evaluate each other's work. They learn from being evaluators, from figuring out how to tell the differences between great work, good work and not-so-good.
- Student presentations provide a good opportunity for feedback. Weak presenters get better with practice.
- Feedback can be friendly and respectful while still pointing out areas for improvement.

# Our teaching approach

## Sample questions

- We draw exams from 100 essay questions listed in a study guide. We think this approach helps students develop their writing skills and their ability to work in groups.

See “[Assessment in the software testing course.](#)”

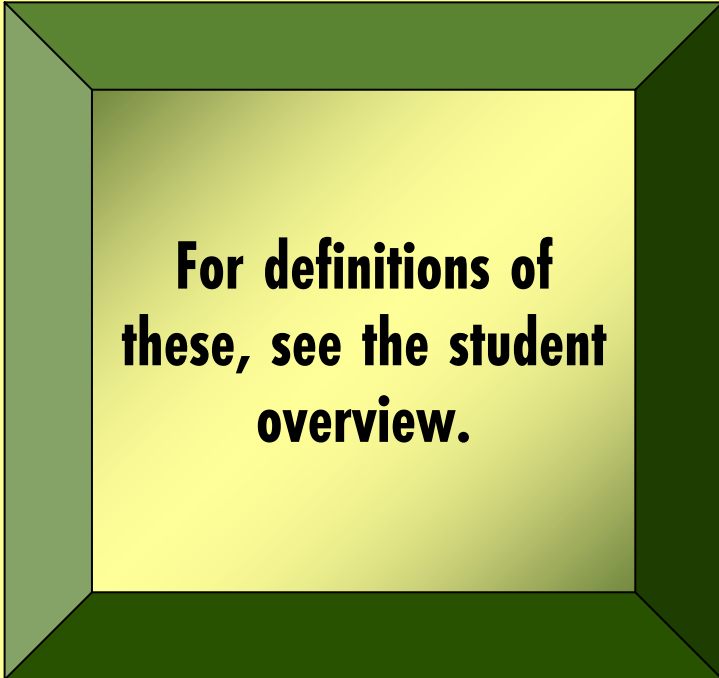
- This works well enough in an academic course, in which students are motivated to pass exams. In an on-the-job study group, we’d pick individual topics for discussion (e.g. create a discussion meeting over lunch once per week) instead.

## Course scope: Key approaches

There are several approaches to testing. We can't cover them all in one course, without making the course too broad / shallow to meet our objectives.

Here are some traditional distinctions:

- Black box versus glass box (white box)
- Behavioral (or functional) versus structural
- Functional versus parafunctional (nonfunctional)
- Unit versus integration versus subsystem versus system
- Verification versus validation
- Acceptance versus independent versus internal black box versus programmer.



**For definitions of these, see the student overview.**

## Scope: Breadth versus Depth

Companies (and their corporate training groups) often emphasize the amount of material covered in a course, rather than the amount learned.

This doesn't work well. The course that races through a huge laundry list of topics won't contribute much to what the students will remember or be able to do.

At Florida Tech, we split black box testing from programmer testing intentionally. The testers rely on different classes of information, apply different skills, identify different bugs on the basis of different evidence.

This doesn't mean we think programmer testing is less important—we require a course in each at Florida Tech. We separate them to avoid teaching a survey course that is too broad and too shallow to be of value.

# The key learning objective

Some people see (and teach) testing as a routine process of straightforward translation of specifications to tests that are run over and over and, and testing “professionalism” as adherence to published standards and “best practices.”

We think that approach leads to dull, shallow courses and weak testing. It’s not what we’re up to.

We see software testing as an investigative process. The tester does **research** to discover information about the quality of the product, **including** research to determine how to effectively test this particular product under the constraints of this particular project. She then reports the results of the research, often to people who wish the results were different.

The foundational objective of this course is to support personal growth of unique individuals as better investigators and communicators.