BBST at AST:
Adaptation of a Course in
Black Box Software Testing

Cem Kaner, J.D., Ph.D.
Florida Institute of Technology

January 2008

Workshop on Teaching Software Testing

These notes are partially based on research that was supported by NSF Grants EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers" and CCLI-0717613 "Adaptation & Implementation of an Activity-Based Online or Hybrid Course in Software Testing." 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.

Software testing:

Empirical, technical investigation of the product under test conducted to provide stakeholders with quality-related information.

Empirical

- We run experiments (tests). Code inspections are valuable, but they are not tests.

technical

- We use technical means, including experimentation, logic, mathematics, models, tools (testing-support programs), and tools (measuring instruments, event generators, etc.)

investigation

- an organized and thorough search for information
- this is an active process of inquiry. We ask hard questions (aka run hard test cases) and look carefully at the results

provide quality-related information

- see next slide (information objectives)

Information Objectives

- Find important bugs, to get them fixed
- Assess the quality of the product
- Help managers make release decisions
- Block premature product releases
- Help predict and control costs of product support
- Check interoperability with other products
- Find safe scenarios for use of the product
- Assess conformance to specifications
- Certify the product meets a particular standard
- Ensure the testing process meets accountability standards
- Minimize the risk of safety-related lawsuits
- Help clients improve product quality & testability
- Help clients improve their processes
- Evaluate the product for a third party

Different objectives require different test techniques and strategies.
They will yield different tests, different test documentation and different test results.

Test techniques: a few examples

- Function testing
- Specification-based testing
- Domain testing
- Risk-based testing
- Scenario testing
- Regression testing
- Stress testing
- User testing
- State-model based testing
- High-volume automated testing

To different degrees, good tests have these attributes:

- **Power**. When a problem exists, the test will reveal it.
- **Valid**. When the test reveals a problem, it is a genuine problem.
- **Value**. It reveals things your clients want to know about the product or project.
- **Credible**. Your client will believe that people will do the things that are done in this test.
- **Representative** of events most likely to be encountered by the user. (xref. Musa's Software Reliability Engineering).
- **Motivating**. Your client will want to fix the problem exposed by this test.
- **Performable**. It can be performed as designed.
- **Maintainable**. Easy to revise in the face of product changes.
- **Repeatable**. It is easy and inexpensive to reuse the test.
- **Pop**. (short for Karl Popper) It reveal things about our basic or critical assumptions.
- **Coverage**. It exercises the product in a way that isn't already taken care of by other tests.
- **Easy to evaluate.**
- **Supports troubleshooting.** Provides useful information for the debugging programmer.
- **Appropriately complex.** As the program gets more stable, you can hit it with more complex tests and more closely simulate use by experienced users.
- **Accountable**. You can explain, justify, and prove you ran it.
- **Cost**. This includes time and effort, as well as direct costs.
- **Opportunity Cost.** Developing and performing this test may prevent you from doing other tests (or other work)

Contexts Vary Across Projects

Testers must learn, for each new product:

- What are the goals and quality criteria for the project
- What skills and resources are available to the project
- What is in the product
- How it could fail
- What the consequences of potential failures could be
- Who might care about which consequence of what failure
- How to trigger a fault that generates the failure we're seeking
- How to recognize failure
- How to decide what result variables to pay attention to
- How to decide what other result variables to pay attention to in the event of intermittent failure
- How to troubleshoot and simplify a failure, so as to better
  - (a) motivate a stakeholder who might advocate for a fix
  - (b) enable a fixer to identify and stomp the bug more quickly
- How to expose, and who to expose to, undelivered benefits, unsatisfied implications, traps, and missed opportunities.

The industrial need for testing courses

- Up to ½ of the software engineering effort involves testing,
- Many companies have 1:5 to 1:1 ratios of testers to programmers

The Test-Related Labor Market

Lots of advice that testers should work as programmers (MS and Google insist on SW engineers in test, rather than black box testers):

- Unit and API test (independent or pair with programmers)
- Write GUI regression test suites
- Write performance tests
- Write test tools
- Write test code to drive devices or other systems
- Write non-regression tests that use technology to reach beyond what humans can do manually,
  - high volume (long sequence) testing
  - high precision testing
  - high diversity (directed search) testing

To what extent should we expect this generation of testers to be programmers?

And for those who are programmers, what should we ask them to do?

Labour Pool

U.S. tech job growth continues

U.S. IT employment continues on a growth path, rising 6% from a year ago to reach 3.68 million employed, according to the most-recent Bureau of Labor Statistics employment survey. IT unemployment was 2%, according to an average of the past four quarters of BLS data, including its most recent third-quarter results. That unemployment rate is down from 2.2% in 2006 and as high as 5.6% in the third quarter of 2003. The total IT workforce, employed and unemployed, also grew about 6% from a year ago. The unemployment rate in management and professional jobs overall was also 2.0%. The biggest job growth categories continue to be software engineers, computer scientists and systems analysts, and IS managers. Software engineers, the largest category, grew 8% from a year ago and make up a quarter of all IT jobs. *(InformationWeek 10/17/07)*

Our Labour Pool – data from 2004

- Nationally, CS enrolment is down 70% since 2001
- **90,000 new software development positions per year** (plus 29,000 support & hw positions).
- **60,000 computing B.Sc. grads**
  - (including computer engineers)
- 20,208 M.Sc. (many have B.Sc. already)
- 40,000 Associate degree (many go on to B.Sc.)
- Many of these are not from the top-ranked universities (2004 data):

– DeVry Institute of Tech	3894	BSCS graduates
– University of Phoenix	2552	
– American Intercontinental	1060	
– Strayer University	993	

Our Labour Pool #3

A CS degree is no guarantee of programming capability. I've visited schools around the country over the past two years.

- Several schools emphasize theory over programming skill (a senior professor at one school told me, "Few of our students can write a working100-line program when they graduate"). This is also widely perceived as a problem common to many CS graduates from India.
- Few CS or Software Engineering programs emphasize soft skills (interviewing, context assessment, usability-oriented design, role playing, persuasive speaking and writing).
- Many courses in design and requirements analysis are essentially tutorials in patterns, UML, and creation of massive template-driven documentation.
- Many courses in software testing are broad and superficial.
- Another block of entrants into the field come from business schools, but many graduates with degrees in "Information Systems" have minimal education in software development or assessment.

Our Labour Pool #4

- My understanding is since 2004:
  - open jobs have increased, while
  - CS enrolment has continued to significantly decrease.
- We appear to have touched bottom and might grow back significantly, but even if enrolment doubles in academic 2008-2009, those folks won't graduate until 2012.

Our Labour Pool #5

What I think this means…

- Of technically proficient graduates interested in testing, most seem to go to big publishers (Microsoft, Google) who aggressively recruit them.
- The IT community is unlikely to meet its needs for new testers with university graduate computer science majors who can write adequate code.

Over the next 5 years, few companies' new-hire testers will be appropriate for test-first programming, glass-box testing or serious test automation.

Labor Pool #6

- Will continue to include large portion of manual testers who have weak backgrounds in computing
  - 40,000 recent certifications by ISTQB
- The question will be how to hire and train the best people for a combination of:
  - Manual testing positions
  - GUI automation positions
  - Non-GUI (e.g. toolbuilder or HVAT) automation positions
  - Glass-box testing and test-first programming positions
- The proportions might shift over time, but the four roles (and in some companies, several other test-group roles) will continue.

Advice I give to employers on who to hire

For much of the past 30 years, many leaders in the testing community have urged us to dumb our work down, make it more routine and then cost-reduce it.

In my view, this often leads to serious inefficiency and weak testing.

Rather than

bringing testing down to a level that weak testers can do it (albeit it, weakly),

I think we should

Hire people with strong potential, and train them to do strong work.

Test groups should offer diverse, collaborating specialists

Test groups need people who understand

- the application under test,
- the technical environment in which it will run (and the associated risks),
- the market (and their expectations, demands, and support needs),
- the architecture and mechanics of tools to support the testing effort,
- and the underlying implementation of the code.

You cannot find all this in any one person. You can build a group of strikingly different people, encourage them to collaborate and cross-train, and assign them to project areas that need what they know.

http://www.kaner.com/pdfs/JobsRev6.pdf

These people need test-related education / training

Academic support for testing

- Few universities teach testing courses
- Many of the newer courses are broad and very shallow (new IEEE/ACM curriculum guide lists 100 pounds of test-content potatoes for a less-than-3-credit sack.)
- Very few universities offer a second / third course in testing

- **Testing is evolving slowly as a field because there is so little educational support for it.**

Commercial training for testing

Commercial short courses are often ineffective because they

- try to cover too much,
- at too shallow a level,
- without application to the learner's specific situation,
- with too little opportunity for practice,
- and less opportunity for assessment and feedback.

Commercial	Academic
Drive-by teaching: 2-5 days, rapid-fire ideas, visiting instructor	Local teaching: Several months, a few hours per week, students get to know instructor
Broad, shallow coverage	Deeper coverage
Time constraints limit activities	Activities expected to develop skills
No time for homework	Extensive homework
No exams	Assessment expected
Coached, repeated practice seen as time-wasting	Coached, repeated practice is highly appreciated
Familiarity	Capability
Work experience helps to bring home concepts	Students have no work experience, need context
Richer grounding in real practice	Harder to connect to real practice
Some (occasional) student groups share a genuine, current need	Students don't naturally come to a course as a group with a shared problem
Objective: one applicable new idea per day	Expect mastery of several concepts and skills

The instructional challenge, as I see it

Software testing

is cognitively complex,

requires critical thinking,

effective communication, and

rapid self-directed learning.

Support for this type of learning requires intense learner engagement, which cannot be met (for most people) in passive-presentation lecture courses

What I'm up to:

- develop courses in an academic environment
  - (where I can learn more about what works and why)
- with the goal of providing an alternative model for commercial (in-house) training and professional self-study

How the academic course works

Students watch video lecture before coming to class

Students often work through an open-book quiz before coming to class

We spend classroom time on

- coached activities
- facilitated discussions
- group feedback (lecture) when I see a class-wide problem

We apply the material in

- in-class activities
- out-of-class assignments

What makes THIS BBST worth the effort?

- BBST combines several ideas about how to teach well
  - I don't think any of the individual ideas are original
  - I think the combination is pretty good
  - I think the following have been success factors for the course:

Success factors in the academic course

1. Strong content
2. Story-based teaching
3. Detailed examples
4. Video lectures
5. In-class activities that tie to the lecture
6. Application to a real product under test
7. Orientation exercises
8. Open book quizzes
9. Study-guide based exam
10. Challenging but focused assignments
11. Task scaffolding
12. Peer review
13. Explicit discussions of learning issues in the course design.
14. Open discussion of (employment) value of the material and the work
15. Organic evolution of the class (rather than process-constrained)
16. Enthusiasm and ongoing renewal (Hawthorne effect)
17. Instruction on test-taking skills
18. Student assessment of learning gains feedback

Should be success factors

I know these should make the course better, but I haven't succeeded in figuring out how:

1. Drill / problem sets, to help students
   - Experience worked examples
   - Develop skills through practice
   - Experience an underlying common core when there is a lot of more superficial variation
2. Paired testing
3. Testing competitions
4. Student presentations
5. Employer / famous-person visitors

Current challenges

1. Disappointing essay exams
2. Getting students to watch videos in advance
3. Getting students to do preparatory exercises
4. Coping with an unstandardizable vocabulary
5. Classroom time management (discussion versus lab time)
6. Videos require significant development time
7. No active discussion in the videos
8. Videos feature one white man
9. On-the-record videos make some storytelling difficult
10. Synchrony is important when students rely on each other
11. Grading time is substantial
12. Activities are hard to design
13. Multiple choice pool is small
14. We need better assigned readings
15. Student prerequisites

SUCCESS FACTORS

Success factors

1. Strong content
2. Story-based teaching
3. Detailed examples
4. Video lectures
5. In-class activities that tie to the lecture
6. Application to a real product under test
7. Orientation exercises
8. Open book quizzes
9. Study-guide based exam
10. Challenging but focused assignments
11. Task scaffolding
12. Peer review
13. Explicit discussions of learning issues in the course design.
14. Instruction on test-taking skills
15. SALG feedback
16. Open discussion of (employment) value of the material and the work
17. Organic evolution of the class (rather than process-constrained)
18. Enthusiasm and ongoing renewal (Hawthorne effect)

Strong content

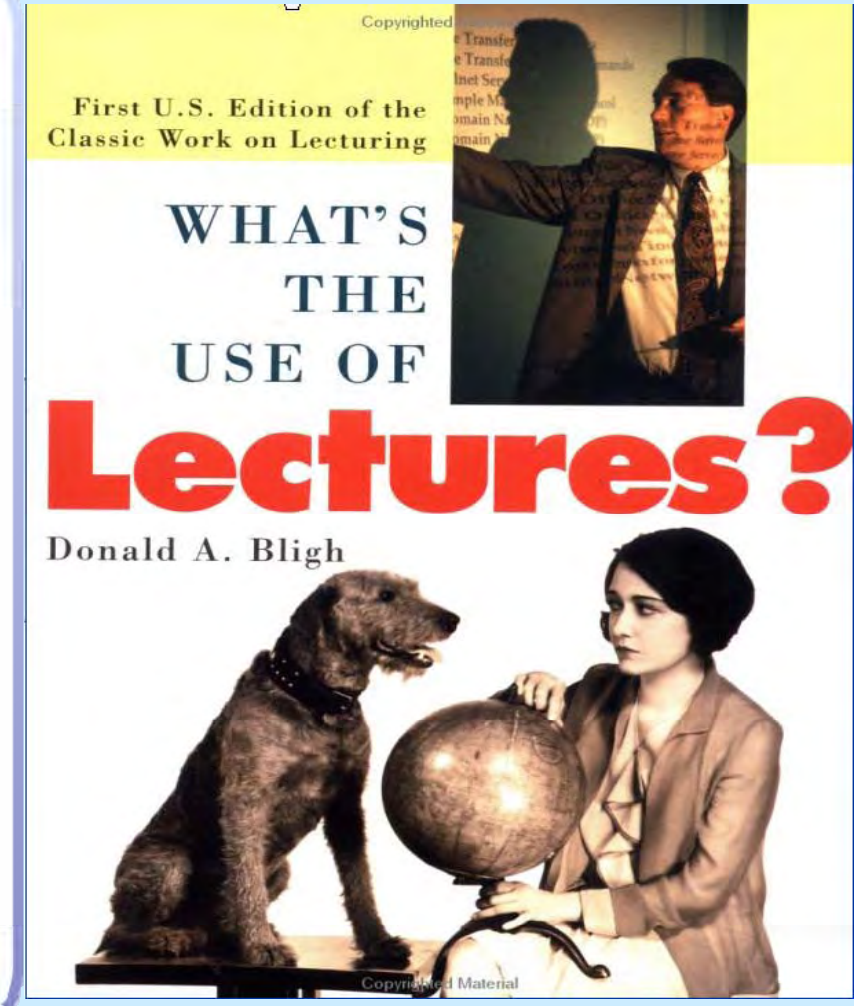
- Course development started in 1983
  - people were teaching QAI/DoD/SWEBOK-style testing rather than Silicon Valley style, totally counterproductive for my staff
  - Outcomes from 1983-1992 were not course notes, they were TCS 1st ed (1987) and 2nd ed (1993) (became best seller in the field)
- Commercial teaching 1993-2004
  - 100+ teachings
  - Extensive peer review (alpha/beta teachings, co-teaching, mergers with other courses)
- Academic 2000-2007

Commercial Teaching Style

Primary communication style was lecture

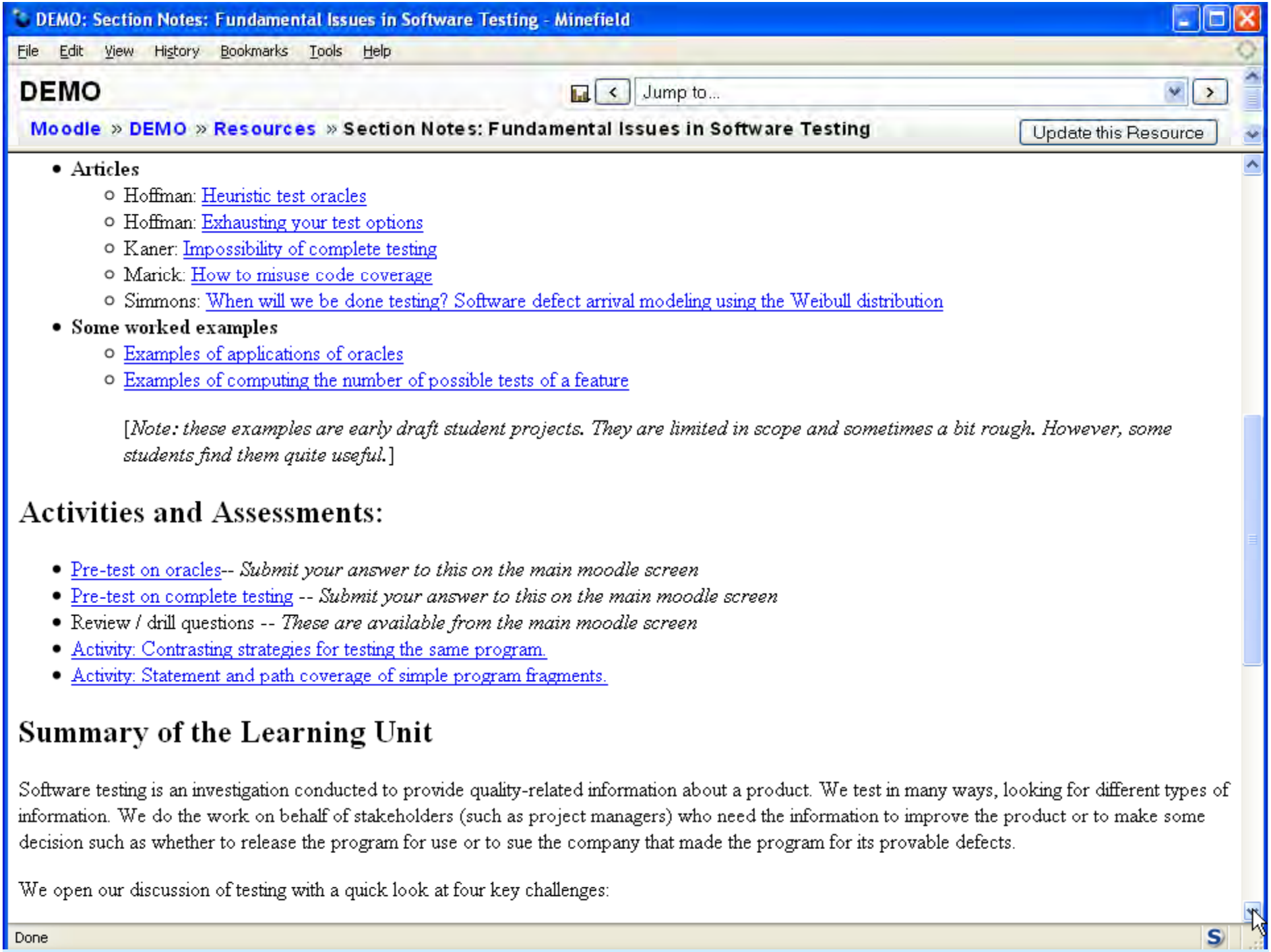
- Real-life examples
  - Motivating
  - Memorable
  - Illustrate applications
  - Illustrate complexity

Lectures can be excellent for conveying basic knowledge, but they are weak for developing higher order cognitive skills



Detailed examples

- For "each" technique, we provide written examples (with screen shots) or video demonstrations that illustrate the application of the technique to a shipping product.
- Worked examples can be powerful teaching tools, especially when motivated by real-life situations. They are fundamental for some learning styles.
- The lasting popularity of problem books, such as the *Schaum's Outline series* and more complex texts like *Sveshnikov [148]* attests to the value of example-driven learning, at least for some learners.
- At this time, we don't have examples for every course section, and what we do have are variable in quality. However, several students have told us the examples that we do provide helped them learn. We intend to create more and better examples.


DEMO: Section Notes: Fundamental Issues in Software Testing - Minefield
File Edit View History Bookmarks Tools Help
DEMO
Jump to...
Moodle » DEMO » Resources » Section Notes: Fundamental Issues in Software Testing
Update this Resource
Articles
Hoffman: Heuristic test oracles
Hoffman: Exhausting your test options
Kaner: Impossibility of complete testing
Marick: How to misuse code coverage
Simmons: When will we be done testing? Software defect arrival modeling using the Weibull distribution
Some worked examples
Examples of applications of oracles
Examples of computing the number of possible tests of a feature
[Note: these examples are early draft student projects. They are limited in scope and sometimes a bit rough. However, some students find them quite useful.]
Activities and Assessments:
Pre-test on oracles-- Submit your answer to this on the main moodle screen
Pre-test on complete testing -- Submit your answer to this on the main moodle screen
Review / drill questions -- These are available from the main moodle screen
Activity: Contrasting strategies for testing the same program.
Activity: Statement and path coverage of simple program fragments.
Summary of the Learning Unit
Software testing is an investigation conducted to provide quality-related information about a product. We test in many ways, looking for different types of information. We do the work on behalf of stakeholders (such as project managers) who need the information to improve the product or to make some decision such as whether to release the program for use or to sue the company that made the program for its provable defects.
We open our discussion of testing with a quick look at four key challenges:
Done

Story-based teaching

- Exemplars (prototypic cases or events) play an important role in the development and recollection of simple and complex concepts.
- Stories are like examples, but they probably also describe something that actually happened and include human-interest information that makes the situation more memorable and more motivating.
- Well chosen stories can enhance the credibility of the teacher with the students and can shape the attitudes of the students.
- Poorly chosen, poorly researched, dishonestly told, or poorly presented stories can kill the credibility of the teacher.

Video lectures

- We created a variety of out-of-classroom activities, such as homework (with application to real products) and group study sessions
- Students praised the (by now, well polished) lectures
- But they often told us that they learned the most from the out-of-class activities
- In many cases, the most effective (our subjective assessment) student-and-instructor interactions happened out of class, such as discussions at the local cafe.
- So we turned the class inside out
  - **Lectures out** of the classroom
  - **Activities** (including discussion) **in** the classroom

Video lectures

- *Stored lectures* are common in distance learning programs.
- Some students prefer live lectures but on average, students learn as well from video as live lecture.
- Students can replay videos which can help students whose first language is not English.
- Web-based lecture segments supplement some computer science courses. Studio-taped, rehearsed lectures with synchronously presented slides (like ours) have been done before.
- Many instructors tape live lectures, but some (including us) report their students prefer studio-produced lectures over recorded live lectures.
- We prefer studio-produced lectures because they have no unscripted interruptions and we can edit them to remove errors and digressions.

Lectures On-Line

http://www.testingeducation.org/BBST

Video lectures

- Students watch them before class
- Take simple quiz that checks that they watched the video and paid attention
- Then we do in-class activities

The results seem good

- Good student satisfaction
- Not enough time for the activities
  - **In an in-house course, time is not constrained by the same type of schedule. It's constrained by value to the project and the staff.**

In-class activities that tie to the lecture

- At Florida Tech, we teach in a lab with one computer per student. Students work in groups. Activities are open book, open web. The teacher moves from group to group asking questions, giving feedback, or offering supplementary readings that relate to the direction taken by an individual group.
- Classroom activities vary. Students might apply ideas, practice skills, try out a test tool, explore ideas from lecture, or debate a question from the study guide.
- Students may present results to the class in the last 15 minutes of the 75-minute class.
- They often hand in work for (sympathetic) grading: we use activity grades to get attention and give feedback, not for high-stakes assessment. We want students laughing together about their mistakes in activities, not mourning their grades
- Developing good activities is sometimes easy, sometimes very difficult. We need to develop a large pool of activities and activity ideas.

Sample Activity: Contrasting Missions

Your group is testing a spreadsheet / database. Please consider what your testing strategy should be and what types of test documentation to deliver.

Different groups consider this question:

- Traditional end-of-cycle test group
- Development support near start of project
- Testing a character database for a game
- Testing a custom application for a medical device maker

Groups report back, either by report/discussion to full group or by rotation of group representatives into discussion groups

Application to a real product under test

- Like service learning, but not as heavy a commitment for the students or for me
  - We pick a well-known product
  - Students apply what they learn to that product
  - Typically, I use an open source product because it avoids NDA problems, students can show their work at interviews
  - Facilitates student learning (application level and above)
  - Facilitates student transfer of skills / knowledge to the workplace because students are doing the same tasks and facing the same problems as with commercial products
  - Work-products (results of assignments) are often credible work examples for employment interviews

Application to a real product

As long as the assignments:

- **are not too far beyond the skill and knowledge level of the learner**,
- authentic assignments yield positive effects on retention, motivation, and transfer [48, 52, 119, 153].

Orientation exercises

This is a special type of classroom activity:

- The task addresses an example or a task or a problem presented in the lecture that the student is about to watch.
- The expectation is that the student will not be able to complete the task correctly before seeing the lecture, but he can make some progress and gain insight into the underlying challenges of the problem.
- The intent is to ready the student to appreciate the solution presented in the lecture:
  - Cognitive readiness
  - Motivational readiness

Open-book quizzes

These count very little toward the final grade, just enough to keep grade-conscious students motivated.

The instructional objectives of these quizzes:

- Help the student notice and understand key concepts and definitions
- Help the student check her understanding of the key concepts and definitions
- Raise an alarm to the student who is reading / watching but not understanding the materials.

Many testing concepts have conflicting definitions or applications. Students are "expected" to know the one from lecture. Quiz discussion forums can help students challenge that one true lecture definition, which builds their knowledge.

Study-guide based exam

- http://www.testingeducation.org/k04/BBSTreviewfall2005.htm
- 100 questions, include all candidates for mid-term and final exam
- Students prepare answers together, assess each other's work
- I can require well-organized, thoughtful answers
- Fosters strategic preparation
- Reduces disadvantage of students whose native language is not English
- Creates cooperative learning tasks that should help limited-English-proficiency students improve language skills

Study guide results

- Students inexperienced with these, often blow the first test
- Make-up mid-terms
  - Replace grade, not average, not best 1 of 2 results
  - Students who take it improve more (1st test compared to final exam) than students who did not take it
    - Practice effect, motivation confound
- Writing is better, answers are better, I have greater freedom to grade less forgivingly
- Many students told me this was the most valuable learning experience in the course, and the most time-consuming

Study Guides

In-house use:

- Focus discussion of course materials
- Potential interview questions, especially if you revise them to apply to your class of product

Challenging but focused assignments

I tend to give 4-6 smaller scale assignments, rather than 1-3 larger scale ones.

Ideally, each assignment would be:

- Linkable to an instructional unit
- Authentic (motivating)
- Peer-reviewable
- Appropriately complex

Task scaffolding

- Scaffolding helps the student understand what is required in a task and how to do it. We provide:
  - Grading rubrics for some assignments
  - Study guides
  - Lecture examples that are similar to the example under test
  - Peer coaching (it's OK to share notes / ideas)

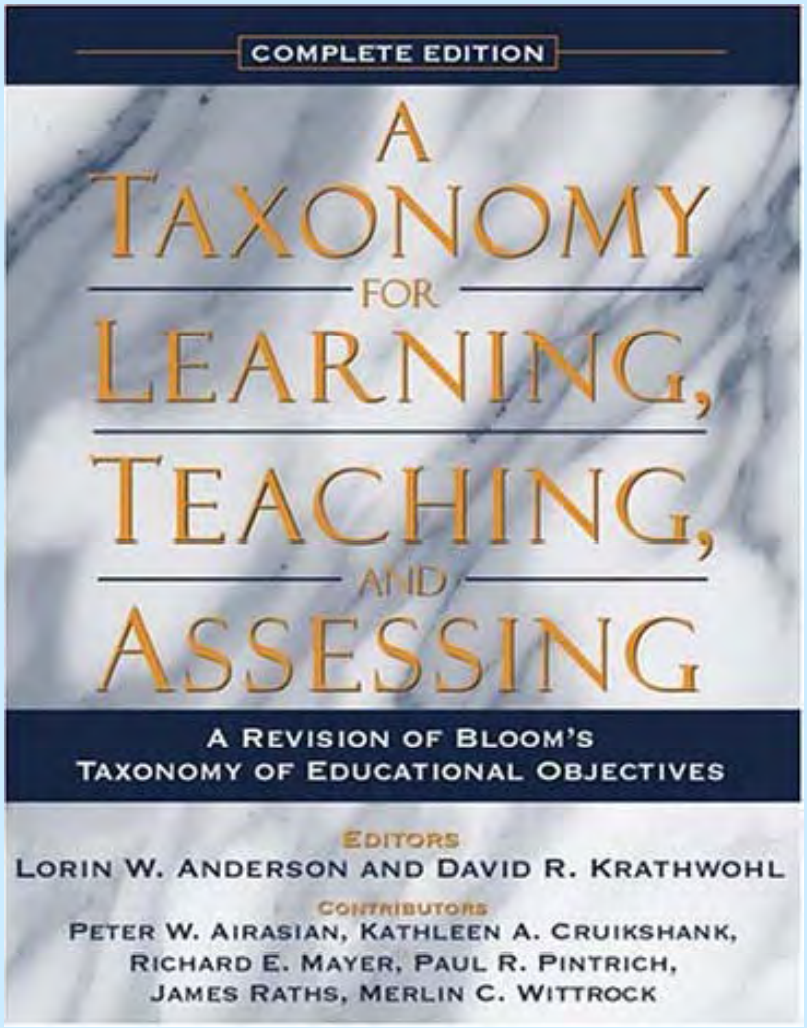
Peer review

- Every assignment
- Every exam
- If you submit something (e.g. for grading), it gets peer-reviewed
- We often provide the reviewers with rubrics to guide their reviewing.

Explicit discussion of learning issues

- I tell people what the learning objectives are for the course
- When we talk about exams, assignments or other tasks, I tie these to the learning objectives
  - I explain that quizzes are open book because the objective of the quiz is to help the student focus her reading, gain familiarity-level understanding of the material. The quiz acts as a study guide.
  - I explain the study-guide exam structure in terms of allowing time to develop higher-level answers, give examples of top-graded answers that acknowledged but then blasted my views. Unlimited time to prepare these..

Characterizing Cognitive Complexity



Anderson & Krathwohl (2001) provide a modern update to Bloom's (1956) taxonomy

Anderson Krathwohl update to Bloom's taxonomy, modified slightly for software testing

Knowledge dimension	Cognitive Process Dimension					
	Remember	Understand	Apply	Analyze	Evaluate	Create
Facts	Lecture	Lecture				
Concepts	Lecture	Lecture				
Procedures	Lecture	Lecture				
Cognitive strategies	Lecture	Lecture				
Models	Lecture	Lecture				
Skills						
Attitudes	Lecture	Lecture				
Metacognition	Lecture	Lecture				

Variation for Testing: Facts

A "statement of fact" is a statement that can be unambiguously proved true or false. For example, "James Bach was born in 1623" is a statement of fact. (But not true, for the James Bach we know and love.) A fact is the subject of a true statement of fact.

Facts include such things as:

- Tidbits about famous people
- Famous examples (the example might also be relevant to a concept, procedure, skill or attitude)
- Items of knowledge about devices (for example, a description of an interoperability problem between two devices)

Variation for Testing: Concepts

A concept is a general idea. "Concepts are abstract in that they omit the differences of things in their extension, treating them as if they were identical." (wikipedia: Concept).

In practical terms, we treat the following kinds of things as "concepts" in this taxonomy:

- definitions
- descriptions of relationships between things
- descriptions of contrasts between things
- description of the idea underlying a practice, process, task, heuristic (whatever)

Here's a distinction that you might find useful.

- Consider the oracle heuristic, "Compare the behavior of this program with a respected competitor and report a bug if this program's behavior seems inconsistent with and possibly worse than the competitor's."
  - If I am merely describing the heuristic, I am giving you a concept.
  - If I tell you to make a decision based on this heuristic, I am giving you a

Sometimes, a rule is a concept.

- A rule is an imperative ("Stop at a red light") or a causal relationship ("Two plus two yields four") or a statement of a norm ("Don't wear undershorts outside of your pants at formal meetings").
- The description / definition of the rule is the concept
- Applying the rule in a straightforward way is application of a concept
- The decision to puzzle through the value or applicability of a rule is in the realm of cognitive strategies.
- The description of a rule in a formalized way is probably a model.

Variation for Testing: Procedures

"Procedures" are algorithms. They include a reproducible set of steps for achieving a goal.

Consider the task of reporting a bug. Imagine that someone has

- broken this task down into subtasks (simplify the steps, look for more general conditions, write a short descriptive summary, etc.)
- and presented the tasks in a sequential order.

This description is intended as a procedure if the author expects you to do all of the steps in exactly this order every time.

This description is a cognitive strategy if it is meant to provide a set of ideas to help you think through what you have to do for a given bug, with the understanding that you may do different things in different orders each time, but find this a useful reference point as you go.

Variation for Testing: Cognitive Strategies

"Cognitive strategies are guiding procedures that students can use to help them complete less-structured tasks such as those in reading comprehension and writing. The concept of cognitive strategies and the research on cognitive strategies represent the third important advance in instruction.

There are some academic tasks that are "well-structured." These tasks can be broken down into a fixed sequence of subtasks and steps that consistently lead to the same goal. The steps are concrete and visible. There is a specific, predictable algorithm that can be followed, one that enables students to obtain the same result each time they perform the algorithmic operations. These well-structured tasks are taught by teaching each step of the algorithm to students. The results of the research on teacher effects are particularly relevant in helping us learn how teach students algorithms they can use to complete well-structured tasks.

In contrast, reading comprehension, writing, and study skills are examples of less- structured tasks -- tasks that cannot be broken down into a fixed sequence of subtasks and steps that consistently and unfailingly lead to the goal. Because these tasks are less-structured and difficult, they have also been called higher-level tasks. These types of tasks do not have the fixed sequence that is part of well-structured tasks. One cannot develop algorithms that students can use to complete these tasks."

Gleefully pilfered from: Barak Rosenshine, Advances in Research on Instruction, Chapter 10 in J.W. Lloyd, E.J. Kameanui, and D. Chard (Eds.) (1997) Issues in educating students with disabilities. Mahwah, N.J.: Lawrence Erlbaum: Pp. 197-221. http://epaa.asu.edu/barak/barak.html

In cognitive strategies, we include:

- heuristics (fallible but useful decision rules)
- guidelines (fallible but common descriptions of how to do things)
- good (rather than "best" practices)

The relationship between cognitive strategies and models:

- deciding to apply a model and figuring out how to apply a model involve cognitive strategies
- deciding to create a model and figuring out how to create models to represent or simplify a problem involve cognitive strategies

BUT

- the model itself is a simplified representation of something, done to give you insight into the thing you are modeling.

We aren't sure that the distinction between models and the use of them is worthwhile, but it seems natural to us so we're making it.

Variation for Testing: Models

A model is

- A simplified representation created to make something easier to understand, manipulate or predict some aspects of the modeled object or system.
- Expression of something we don't understand in terms of something we (think we) understand.

A state-machine representation of a program is a model.

Deciding to use a state-machine representation of a program as a vehicle for generating tests is a cognitive strategy.

Slavishly following someone's step-by-step catalog of best practices for generating a state- machine model of a program in order to derive scripted test cases for some fool to follow is a procedure.

This definition of a model is a concept.

The assertion that Harry Robinson publishes papers on software testing and models is a statement of fact.

Sometimes, a rule is a model.

- A rule is an imperative ("Stop at a red light") or a causal relationship ("Two plus two yields four") or a statement of a norm ("Don't wear undershorts outside of your pants at formal meetings").
- A description / definition of the rule is probably a concept
- A symbolic or generalized description of a rule is probably a model.

Variation for Testing: Skills

Skills are things that improve with practice.

- Effective bug report writing is a skill, and includes several other skills.
- Taking a visible failure and varying your test conditions until you find a simpler set of conditions that yields the same failure is skilled work. You get better at this type of thing over time.

Entries into this section will often be triggered by examples (in instructional materials) that demonstrate skilled work, like "Here's how I use this technique" or "Here's how I found that bug."

The "here's how" might be classed as a:

- procedure
- cognitive strategy, or
- skill

In many cases, it would be accurate and useful to class it as both a skill and a cognitive strategy.

Variation for Testing: Attitudes

"An attitude is a persisting state that modifies an individual's choices of action." Robert M. Gagne, Leslie J. Briggs & Walter W. Wager (1992) "Principles of Instructional Design" (4th Ed),, p. 48.

Attitudes are often based on beliefs (a belief is a proposition that is held as true whether it has been verified true or not).

Instructional materials often attempt to influence the student's attitudes.

For example, when we teach students that complete testing is impossible, we might spin the information in different ways to influence student attitudes toward their work:

- given the impossibility, testers must be creative and must actively consider what they can do at each moment that will yield the highest informational return for their project
- given the impossibility, testers must conform to the carefully agreed procedures because these reflect agreements reached among the key stakeholders rather than diverting their time to the infinity of interesting alternatives

Attitudes are extremely controversial in our field and refusal to acknowledge legitimate differences (or even the existence of differences) has been the source of a great deal of ill will.

In general, if we identify an attitude or an attitude-related belief as something to include as an assessable item, we should expect to create questions that:

- define the item without requiring the examinee to agree that it is true or valid
- contrast it with a widely accepted alternative, without requiring the examinee to agree that it is better or preferable to the alternative
- adopt it as the One True View, but with discussion notes that reference the controversy about this belief or attitude and make clear that this item will be accepted for some exams and bounced out of others.

Variation for Testing: Metacognition

Metacognition refers to the executive process that is involved in such tasks as:

- planning (such as choosing which procedure or cognitive strategy to adopt for a specific task)
- estimating how long it will take (or at least, deciding to estimate and figuring out what skill / procedure / slave-labor to apply to obtain that information)
- monitoring how well you are applying the procedure or strategy
- remembering a definition or realizing that you don't remember it and rooting through Google for an adequate substitute

Much of context-driven testing involves metacognitive questions:

- which test technique would be most useful for exposing what information that would be of what interest to who?
- what areas are most critical to test next, in the face of this information about risks, stakeholder priorities, available skills, available resources?

Questions / issues that should get you thinking about metacognition are:

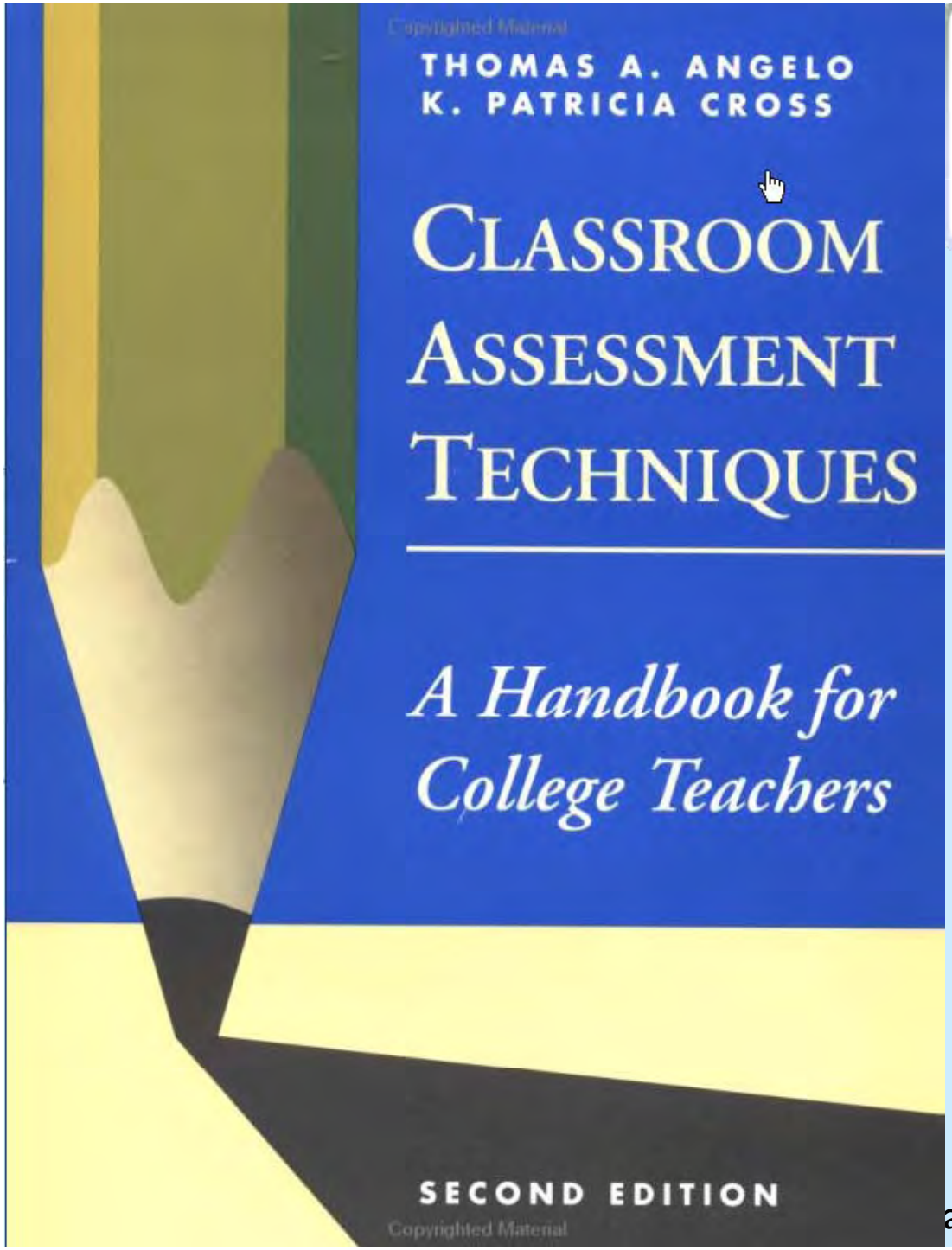
- How to think about ...
- How to learn about ...
- How to talk about ...

In the BBST course, the section on specification analysis includes a long metacognitive digression into active reading and strategies for getting good information value from the specification fragments you encounter, search for, or create.

Assessment

1. Assessment at one level (e.g. facts / concepts) is not informative with respect to another level (e.g. evaluation)
2. "Authentic assessment" – assessment with simplified or artificial tasks is uninformative with respect to what can actually be done in real circumstances (flip side of the transfer problem)
3. Assessment that is apparently at a higher level is often reducible to lower level via:
    1. Study strategies
    2. Question-answering strategies

    This is part of the strong success of exam-review courses.



We set objectives

- Cognitively complex material
- We need to develop skill, judgment, and attitudes, not just knowledge of facts and definitions
- We face the usual (for science education) transfer problems
- Set a few explicit learning objectives
- And assess against them

My Learning Objectives

- Learn many test techniques well enough to know how, when, and why to use them
- Foster strategic thinking--prioritization, designing tests/reports for specific audiences , assess the requirements for complex testing tasks (such as test automation, test documentation)
- Apply (and further develop) communication skills (e.g. for bug reporting, status reporting, specification analysis)
- Improve and apply teamwork skills (peer reviews, paired testing, shared analysis of challenging problems)
- Gain (and document) experiences that can improve the student's chances of getting a job in testing

Instruction on test-taking skills

Individual guidance to students on how to do specific tasks involved in a test, such as:

- What is the "call of the question"?
- Separating relevant and incidental (confuser) facts
- Ways of critically reading multiple-choice questions
- How to organize essay answers

We also provide a course video that shows the grading of several student answers' to a complex essay question and explains:

- how we decompose the question into gradable components
- how we grade these simpler parts of the question.

SALG feedback

- Student Assessment of Learning Gains
  http://www.flaguide.org/cat/salg/
- Measures student perceptions of their 'gains' in learning
- Customizable, we customize it heavily
- Administered online
  - We find it easier to summarize data if we use our own form and Survey Monkey, rather than the original site
- FREE
- Beats the standard course evaluation form!
- Students each spent over an hour providing their evaluation.

Employment value of the material and the work

We specifically advise students to create course portfolios that show off the workproducts from this or other courses.

When an assignment has been particularly practical, or has intrigued hiring managers in the past,

- We point this out to students
- We discuss what aspects of the work might be more relevant to the employers

Organic evolution of the class

- Course development is not driven by an inflexible heavyweight process.
- It is driven by what is most valuable, most capable, of allowing imports and expertises.

Enthusiasm and ongoing renewal (Hawthorne effect)

- Hawthorne effect: workers performed better and were more enthusiastic, in response to the combination of (a) change and (b) management attention.
- Usually discussed as an experimenters' trap (experimenter effects)
- Rather than seeing this as a blocking problem, we take advantage it, describe the changes we're making, show that we're interested in how they handle the changed cases, etc.

Opportunities for Improvement

SHOULD-BE SUCCESS FACTORS

Should be success factors

I know these should make the course better, but I haven't succeeded in figuring out how:

1. Drill / problem sets, to help students
   - Experience worked examples
   - Develop skills through practice
   - Experience an underlying common core when there is a lot of more superficial variation
2. Paired testing
3. Testing competitions
4. Student presentations
5. Employer / famous-person visitors

Drill / problem sets

We want to help students:

- Experience worked examples
- Develop skills through practice
- Experience an underlying common core when there is a lot of more superficial variation

Example: Domain Testing

Most widely taught testing technique

- For details, see http://www.testingeducation.org/BBST/Domain.html
- Easy to explain the basic concepts
- Classic examples widely taught
- Students quickly signal that they understand it
- But when you give them exercises under slightly new circumstances
  - They blow it
    - And then they blow the next one
      - And the next one . . .

Common errors

Consider an integer that can take on values from -999 to 999 inclusively

- **Doesn't spot a boundary.**
- **Offers excess values.** Students offer 998 as well as the appropriate 999 and 1000.
- **Doesn't spot a dimension.** (a) how many characters should this field handle? Same for positive and negative numbers? (b) if you delay after entering the first character, is there a risk of time-out? What delay durations should you test? Boundaries?
- **Doesn't articulate a risk.** Suppose we explicitly ask students to identify a risk and then identify relevant variable(s) and a powerful test appropriate to the risk. Rather than describe how the program might fail, the student might reiterate the test or make vague statements, like "fail to process this value correctly."
- **Doesn't explain how a test case relates to a stated risk.** When an assignment calls for such an explanation, students may respond inarticulately or irrelevantly.
- **Doesn't consider a consequence.** In real life (and in some of our test questions), the tester can determine more information than the bare range of an input field. The program will do something with the data entered. It is important, for each of those uses, to check whether the bounds imposed by the input filter are appropriate to the later use, and what consequence will result if they are not.
- **Poor generalization.** In more complex questions than the integer example here, students often pick inappropriate variables for analysis, such as treating each value of a binary variable as the best representative of its own 1-member class.

Common errors

Students have learned the basic idea

- Bloom's taxonomy lower levels: know / explain

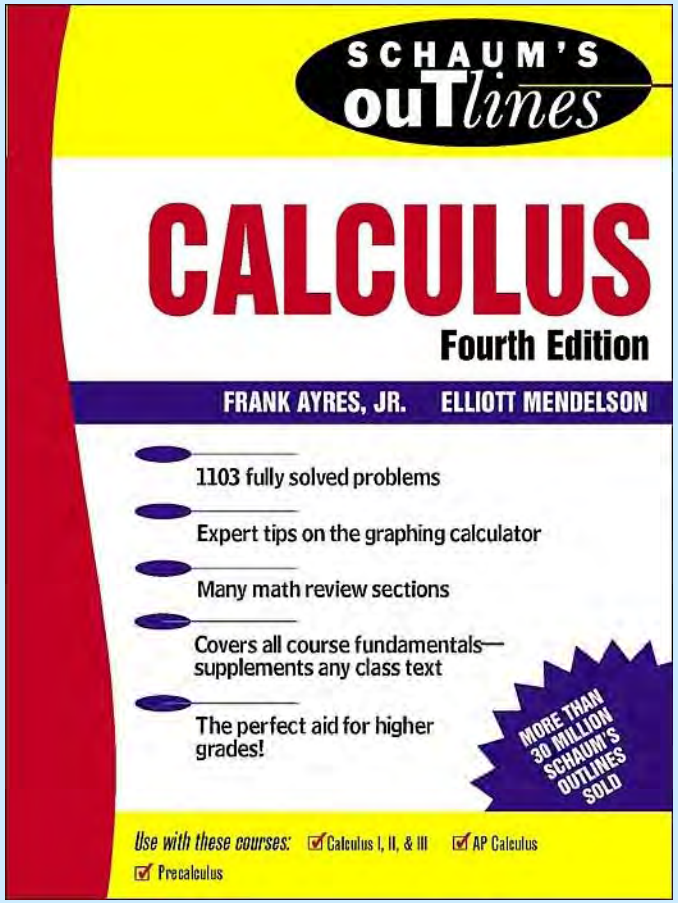
Students don't have a higher-level understanding

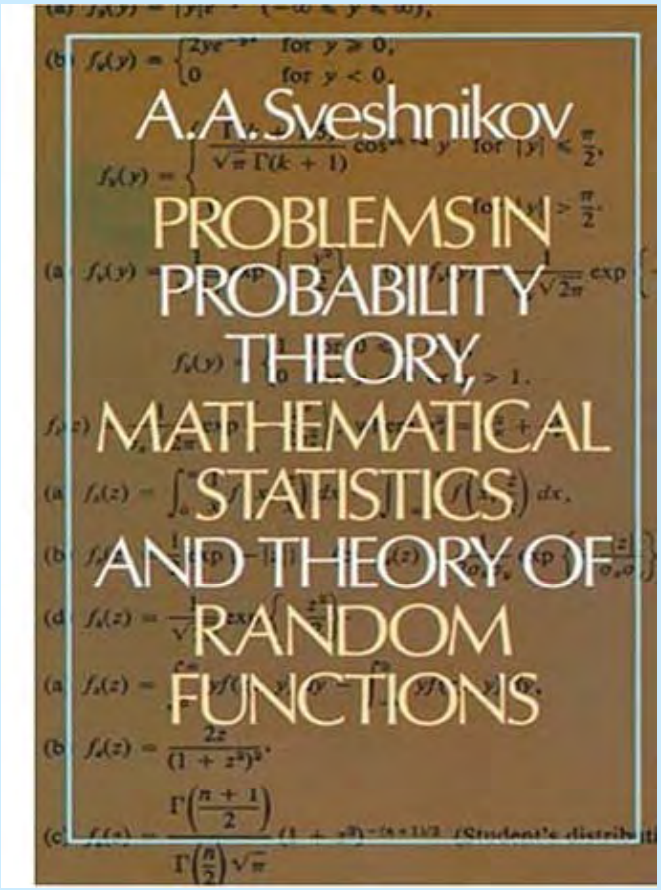
- apply / analyze / think through what they are learning

How can we increase their depth of understanding?

Analogy to studying mathematics

Lots of practice exercises, like we used to do (and love, of course) as math students.





I Tried This With Commercial Students

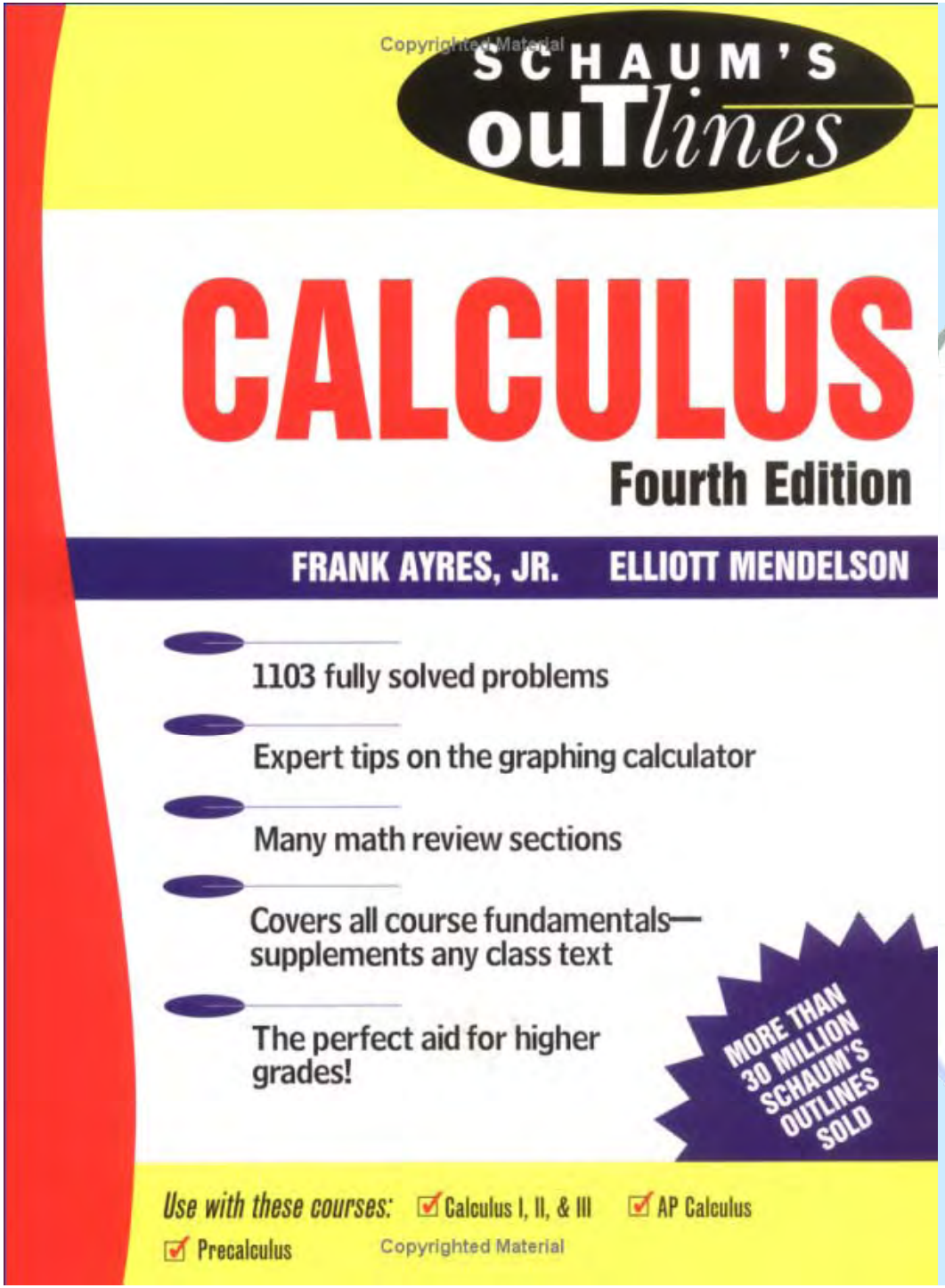
Many (often, most) of them needed a lot of practice under changing circumstances

But the perceived slow pace of the course made them anxious

After 16 years of sending my staff to training, training my own staff, and training strangers . . .

. . . I realized two things:

1. This wasn't working (not for me, not for the field)
2. In terms of commercial training, I didn't know how to make it work



Now, at Florida Tech!

Academic course

- Lots of practice exercises
- Like we used to do as math students
- It was impractical in commercial training
- **Now, at last, we can try it on university students.**

Padmanabhan's Thesis: Practice on Domain Testing

- 18 classroom hours of lecture plus examples plus practice, practice, practice. Lots of procedural instruction and drill
- Students mastered every procedure
- Final exam

– Applied what they knew to similar questions (near transfer)

➢ **They aced them**

– Applied what they knew to a problem that was beyond their practice (not beyond the lecture) (a little bit farther transfer)

➢ **They all failed miserably**

- Successful transfer of learning requires more than procedural training and practice **(Of course, YOU already know that ...)**

Paired testing

- Two-person group projects:
  - File bug reports together
  - Edit each other's reports
  - Split tasks so that each plays 1 or (a very few) roles.
- Bug-hunting competitions

- In practice, I've seen remarkable resistance to this, and little benefit to few students.

Testing competitions

Students create test-related work products (bug reports, etc.) under time pressure.

How well / how much do they actually do?

Student presentations

- 5 – 15 minutes
- Learn a lot about a topic by teaching it
- The other students in the room are typically bored, sometimes rude

Employer / famous person visits

- Good stories (memorable examples)
- Can provide authoritative answers
- Can credibly resolve some differences of opinion among students (visitor must have involvement with the issue under dispute)
- Enhance the reputation of the instructor (cool to be connected to these people)
- Opportunity to create Hawthorne-effect application for the course.