Running Head: CS Course Makeover

Inside Out: A Computer Science Course Gets a Makeover

Cem Kaner[1*], Rebecca L. Fiedler[2**]
[1]Florida Institute of Technology, [2]University of Central Florida

This work was partially 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 authors and do not necessarily reflect the views of the National Science Foundation. The authors would also like to acknowledge the collaboration of James Bach, Hung Quoc Nguyen, and Doug Hoffman in the development of the underlying course materials and the encouragement and support of Dr. William Shoaff.

Correspondence concerning this paper should be addressed to Cem Kaner, Computer Science Department, Florida Institute of Technology, Melbourne, Florida 32901, E-mail: kaner@kaner.com

Inside Out: A Computer Science Course Gets a Makeover

Abstract: This paper outlines the radical restructuring of a Computer Science course on Software Testing. Authors describe the course's evolution from a predominantly lecture-based format to one based on projects and in-class activities. This was accomplished by adding videotaped lectures students watch outside of class. All course materials are available under a Creative Commons license to academics, commercial trainers, and self-studiers around the world.

Software testing is a relatively narrow subfield of software engineering, but many of the challenges we've faced in teaching it are commonplace in science, technology, engineering, and mathematics (STEM) education. We've developed an instructional approach that is working well in our academic setting and that we believe can extend to other STEM courses at university and to commercial training.

Still a work in progress, the current version of the course differs from the traditional lecture course in several ways:

- We moved lectures out of the classroom. We tape them in advance and post them on a website. Students watch lectures before coming to class. Frequent low-stakes quizzes motivate students to keep pace and watch the videos.
- Classroom contact hours are for coached activities, discussions, and student presentations.
- Students apply the lessons of the course to a well-known program, typically an open-source program in development. We use the same program throughout the term: students join its development team and apply the skills and knowledge they gain in the course in ways that help the project. The artifacts they create are often useful later, in job interviews. This offers many of the same benefits of service learning, but in a way that is logistically simpler and less demanding for the instructor than some service learning projects.
- Students are incentivized, but not required, to work in groups on all types of homework. Students write examinations on their own, but most students prepare for exams in teams.
- We draw exam questions from a published pool, handed out as a study guide. Students prepare for exams by writing answers to the study guide questions. Because students have had plenty of open-book preparation time and opportunity for peer-review, we can require more carefully considered, better written answers.
- The course materials are available to the public (at http://www.testingeducation.org/BBST) under a Creative Commons license. Other university and commercial instructors use them in their courses, providing feedback and additional content.

This report describes the evolution of this course, lessons learned, and progress to date.

The Commercial Course

The course started as in-house training based on *Testing Computer Software* (Kaner, 1988), which evolved into the best selling book in the field (Kaner, Falk, & Nguyen, 1999). Kaner and Nguyen developed a public version, initially offering it to working software testers and career-changers through the American Society for Quality and University of California (Berkeley) Extension in 1993/1994. That course was reviewed, rewritten, and extended by several colleagues, some of them co-authors (Kaner,

Bach, & Pettichord, 2001; Kaner, Falk, & Nguyen, 1993). The course matured into a commercially successful practitioner-training course. By 2000, Kaner had taught it over 100 times, many of those at leading software companies. Nguyen, his staff, and other licensees, taught it many times more.

Commercial client expectations of courses like these are limited. Clients, fellow trainers and consulting companies who sell training services have repeatedly advised us that clients will be satisfied if each staff member in training gains one idea s/he considers useful from a three-day course, and very satisfied with one useful idea per day.

Clients expected lecture-style instruction that covered a wide range of ideas and techniques at a crisp pace. Despite the popularity of this style of course, full-day lectures are not well-suited to helping students develop new skills or develop a deep enough understanding of the material to assess what testing technique is most likely to be effective under which circumstances (Bligh, 2000). The style and the time compression flood the students with information and allow little time for individual practice or reflection. Kaner introduced some activities into his courses—brief, simple activities were well received, but there was often client resistance to tasks that were more challenging or clusters of tasks designed to approach a technique from several angles to better support transfer from classroom examples to workplace application. These activities had to be done in class—students often have to catch up on their other work at night and don't have time to do homework in short full-day courses. The more time on in-class activities, the fewer topics can be covered.

The Academic Course

In 1999, Kaner accepted a professorship at Florida Institute of Technology, with the goal of deepening software testing education beyond what can be achieved with practitioner-level training. As part of this work, he founded Florida Tech's *Center for Software Testing Education & Research* (http://www.testingeducation.org). One of the Center's projects has been transformation of the introductory software testing course from commercial to academic. From there, we are developing materials useful for other university instructors, commercial instructors, and self-studiers. We (Kaner and Fiedler) began collaborating on this project in 2001.

University courses differ from commercial courses in key ways:

- The course lasts over a longer period of time, introducing material at a much slower pace (a few hours per week), with plenty of time for practice and reflection outside the classroom.
- Students expect to do homework.
- Students expect the instructor to assess their work and provide detailed feedback on their performance.
- Frequent assessments give the instructor objective feedback on the quality and effectiveness of instruction as well as insight into the knowledge of the students.
- Students generally welcome coached practice in class and appreciate it when the instructor takes extra time to make sure they are mastering key concepts in the course. They are more interested in the congruence between the level of support provided in class and the level of emphasis on the exam than on the length of the checklist of topics covered.

Kaner typically teaches the introductory testing course at Florida Tech twice per year. The opportunity to assess student and instructor performance enabled an assessment-driven evolution of the course from 2000 to 2005. Our key questions were:

- What is the broad scope or subject matter of the course?
- What do we want to achieve in the course? (What are our learning objectives?)

- How will we assess the students' knowledge and progress?
- What learning experiences will better prepare students for these assessments and for the workforce?

Instructional theory books often present these as sequential questions. In practice, we evolved the course on all three dimensions in parallel.

Subject Matter

To put the instructional approach in context, we start with a few details about the craft of software testing and the challenges doing and teaching it.

We see software testing as *a process of technical investigation of the product under test conducted to provide stakeholders with quality-related information.*

The classic definition of software testing is "the process of executing a program with the intent of finding errors" (Burnstein, 2003; IEEE, 1983; Kaner, Falk, & Nguyen, 1999; Myers, 1979, p. 5). Another second classic definition is evaluation of the product (IEEE, 1990). A third, helping the project manager decide when to put a product into production or release it to the public (Marick, 1997). There are several others (Kaner, Bach, & Pettichord, 2001). The common thread among the definitions is that software testers gather factual information about the product's quality on behalf of their (in-house or external) clients.

Testing is often focused on the external behavior of the program, without reference to the underlying code. This is called *black box testing*. In contrast, *white box* or *glass box* tests are designed on the basis of the implementation details of the code. One class of glass box testing is *programmer testing*, testing done by the programmer of her or his own code.

In deciding the curriculum for our recently-accredited Bachelor's degree in Software Engineering, Florida Tech decided to require students to complete two testing courses, the first focused on black box testing, the second applying the first course's lessons in test design and critical thinking to programmer testing.

The course described in this report is the black box testing introduction.

Black box testing involves a cognitively complex collection of tasks. Here are a few of the challenges:

- For any nontrivial program, the population of possible tests is vast. A common example of this presents a simple, well-structured program that runs about 20 lines but requires 100 trillion tests. At five minutes per test, running all of these tests would take a billion years (Myers, 1979). Along with running tests, testers must interpret the results, troubleshoot and report failures, document their work, provide assessments and other reports, and so on. These are all worthy tasks but they are unachievable in the time available. Therefore, testers must prioritize ruthlessly, choosing which worthwhile tasks to skip or drastically subset.
- Another problem that is surprisingly rarely discussed in schools and process improvement meetings is that it is very often very challenging to decide whether the program actually passed or failed a test. The task is simplified when there is a complete, authoritative specification but these are rare in commercial development. Additionally, every test is complex. Consider this example. Suppose you test a program that adds two numbers, and write an automated test that feeds the program under test the numbers and checks the sum. Feed the program 2 and 3, the result is 5. This looks like a passed test, but suppose that it took the program 10 hours to compute the sum—that would be unacceptable, but the automated test tool might well pass the test (correct sum) without

noticing the excessive time. Every test has many possible results (not just the sum and the time). It can change memory, change the state of connected devices, send messages, and so on. A leading office automation product manufacturer showed Kaner the diagnostics in its firmware. The company's staff run diagnostics after every test when doing some types of automated testing. The challenge is that there are 1100 different diagnostics. It is impractical to check them all after any test: it takes too much time and running the diagnostics changes the state of the device under test. Thus any test can appear to pass by some measure even if it actually fails. We've also seen examples in which the program appears to fail a test, but is actually behaving correctly. Assessing the results of a test—especially deciding what to assess—requires judgment.

- There is also broad disagreement about how to measure and report testing progress (Kaner, 2001; Kaner & Bond, 2004).

Testers must choose and apply methods appropriate to their goals and feasible within their project's constraints. The vast population of possible tests takes the cost of complete testing beyond even the broadest imaginable constraints *and therefore* a key aspect of testing work is prioritization among worthy tasks that cannot all be done. Another key testing task is communicating investigative results, such as bug reports, project status reports, and product assessments. These reports can be controversial, may be taken as personal criticism, and especially for bug reports, must be so clearly written that someone else can do detailed technical work on the basis of them. In the course of their work, testers create products of their own, such as suites of automated tests (significant bodies of code in their own right) and test documentation (which may run thousands of pages). Testers must be able to determine their own and their stakeholders' requirements for such work products.

Learning Objectives

We were guided in developing learning objectives by Angelo & Cross' (1993) *Teaching Goals Inventory*. The course changed format several times, but the objectives have been fairly stable.

- Students should learn a variety of testing techniques and a variety of dimensions along which they differ—and to understand the variation well enough to decide which technique is more promising for a given situation, why, and how to apply it.
- Students should apply (and further develop) communication skills, for example in appraising and writing effective bug reports, reporting status, and analyzing specifications.
- Students should gain practical experience and create artifacts that can help them demonstrate competence in employment interviews.
- Students should work in teams, developing and presenting material to each other and cross-checking each others' work. Collaborative work is common in industry (Bransford, Brown, & Cocking, 2000) and particularly common for industrial testers, whose primary work products are documents that influence others to act (fix bugs, change schedules, change designs, etc.) and who often review each other's work for impact, clarity, and appropriateness of emotional tone.
- Students should apply strategic thinking, including prioritizing tasks, designing tests and reports for specific audiences, and assessing requirements for complex testing tasks.

Presentation of the Content

The primary strength of the commercial course was a polished set of lectures with many real-life anecdotes. According to Bligh's (2000) summary of the literature on lecture effectiveness, lectures can be as effective as other instructional techniques for transmitting basic information about a topic, but they are less effective than some other methods for teaching behavioral skills, promoting higher-level thinking, or

changing attitudes or values. In terms of Anderson et al's (2001) learning objective taxonomy, lectures would be most appropriate for conveying factual and conceptual knowledge at the remembering and understanding levels. Our students need to learn the material at these levels, but as part of the process of learning how to analyze situations and problems, apply techniques, and evaluate their own work and the work of their peers.

Carefully crafted lectures offer several other benefits. Experienced, charismatic lecturers can share examples to help students learn complex concepts, tasks, or cultural norms (Ford, 2002; Hamer, 1999; Kaufman & Bristol, 2001).

Lecturers can convey the *lecturer's* enthusiasm for the subject, which improves student satisfaction with the course (Williams & Ware, 1977). They can also organize and integrate a complex set of material. As Forsyth (2003, p. 50) eloquently puts it, "Lectures provide the scholar with the means of not just disseminating information but also transforming that information into a coherent, memorable package. Scholars, when they reveal their unique interpretation of questions they have spent years researching and contemplating, are an unmatched source of information and interpretation."

Rather than offering live lectures, we tape and publish them on the course website. This in itself is hardly new. It is common to offer stored lectures as part of a distance learning program (Rossman, 1999). Web-based lecture segments are being used to supplement Computer Science courses (Fintan, 2000). Studio-taped, rehearsed lectures with synchronously presented slides (as we do) have been done and described elsewhere (e.g., Dannenberg).

Students commonly report that they prefer live lectures (Firstman, 1983; Maki & Maki, 2002). However, it appears that, on average, students learn as well from video as from live lecture (Bligh, 2000; Saba, 2003). (We say *on average* because there appear to be individual differences associated with learning style or skill and there are also major effects of online interaction on distance learning not relevant to the live course model we're working with.)

Faculty often comment that live lectures allow them to interact with students in ways that taped lectures do not. We feel the same way, but (a) in practice, relatively few of our students make comments or ask questions in class and (b) we are sometimes surprised by our assessment results of what students failed to learn from some of our lectures. It is true that live lectures allow more interaction than taped lectures viewable on the web, but the non-lecture activities we can develop in our classrooms provide far more time and opportunity for interaction than live lectures.

- Live lectures also present some disadvantages that can be addressed in stored lectures (lectures that can be accessed by students at any time):
- Short-term memory capacity varies among learners, especially as they age. Live lectures can't be rewound: what was said is in the past; if it is not remembered, it is lost (Langer, 2002). In contrast, students watching a recorded video can—and do (He, Gupta, White, & Grudin, 1998)—jump backwards in the video to review what was said.
- Students whose first language is not English often have difficulty keeping up with live lectures and can also benefit from the ability to replay material.
- Live lectures can become fragmented as students ask questions (and are responded to). Not all student questions are informative for everyone, and not all questions come at the most opportune time. In the stored lectures we currently create, there are no students on tape and thus no interruptions.
- Live lecturers sometimes drift or ramble. Stories are not always directly relevant and some take so long that students lose the message. Video lectures can be partially scripted (with a teleprompter), with departures from the script to add spontaneous remarks. An

example that takes too long can be cut during post-production, or shortened in a retelling / retaping of the example.

- Students can replay a stored lecture while studying for an exam or working on an assignment, reviewing the material in the context of a specific problem or task they are working with.

In summary, there are advantages and disadvantages to taped versus live lectures. At this time, we are not convinced that one is better than the other. However, by moving lectures out of class, we create a semester of space for non-lecture interaction with our students. By making the lectures part of the homework, we effectively double the teaching time available to us in the course.

We supplement the lectures with worked examples on the website and required readings. The examples typically show detailed application of a test technique to a published software product. Worked examples can be powerful teaching tools (Clark & Mayer, 2003) especially when they are motivated by real-life situations. They are fundamental for some learning styles (Felder & Brent, 1996). Exemplars play an important role in the development and recollection of simple and complex concepts (Brooks, 1978; Medin & Schaffer, 1978; Smith, 2005). The lasting popularity of problem books, such as the Schaums Outline series, attests to the value of example-driven learning, at least for some learners. At the moment, our examples are pure, non-interactive text. We intend to change these to narrated videos that show the test and the failure in a (taped) real-time demonstration.

The required readings include the course texts (currently Kaner, Bach, & Pettichord, 2001; Whittaker, 2002) and several published articles. We link to the articles on our website when possible, but distribute others at a Florida Tech specific, password-protected site that restricts distribution of the copyrighted articles to Florida Tech students registered for the class.

Assessment and Learning Activities

Anderson et al. (2001) published an extensive revision and update of Bloom's (1956) taxonomy. Their taxonomy is two-dimensional, with *knowledge* and *cognitive processing* dimensions.

- On the *Knowledge Dimension,* the levels are *Factual Knowledge* (such as the definition of a software testing technique), *Conceptual Knowledge* (such as the theoretical model that predicts that a given test technique is useful for finding certain kinds of bugs), *Procedural Knowledge* (how to apply the technique), and *Metacognitive Knowledge* (example: the tester decides to study new techniques on realizing that the ones s/he currently knows don't apply well to the current situation.)
- On the *Cognitive Process* dimension, the levels are *Remembering* (such as remembering the name of a software test technique that is described to you), *Understanding* (such as being able to describe a technique and compare it with another one), *Applying* (actually doing the technique), *Analyzing* (from a description of a case in which a test technique was used to find a bug, being able to strip away the irrelevant facts and describe what technique was used and how), *Evaluating* (such as determining whether a technique was applied well and defending the answer), and *Creating* (such as designing a new type of test.).

For most of the material in the testing course, we want students to be able to explain it (conceptual knowledge, remembering, understanding), apply it (procedural knowledge, application), explain why their application is a good illustration of how this technique or method should be applied (understanding, application, evaluation), and explain why they would use this technique instead of some other (analysis).

One of the key contributions of taxonomies like Bloom's is that they focus us on assessment techniques that are appropriate to the instructional objective.

To help our students learn at a variety of levels, we had to support that learning with a clearly-signaled variety of assessment techniques. These include review quizzes, study-guide driven examinations, applications to the product under test through homework and in-class activities, and additional classroom activities such as orientation exercises and structured discussions.

Review quizzes. Review quizzes are simple objective tests (such as multiple choice or fill-in-the-blank). The typical quiz has ten questions. A student who watches the video before class should pass each quiz. We give a quiz when we want to remind students to watch the lectures before coming to class or when we want to focus students' attention on individual details, such as key definitions. The quiz-review discussion after the test creates a natural opportunity for discussion of the details of interest.

Study-guide driven examinations. At the start of the course, we give students a list of 100 questions. All midterm and final exam questions are drawn from this pool. We have sometimes used the same approach in the computer law, ethics & society course and the metrics & modeling course. The Fall 2005 list is at http://www.testingeducation.org/k04/BBSTreviewfall2005.htm. The full pool that we draw sets of 100 from is at http://www.testingeducation.org/k04/EssayExam.htm and is frequently updated.

In previous iterations of the course, about half of the questions called for definitions, the other half for short or long essay questions. We are now experimenting with a lower proportion of definitions, relying on the quizzes to focus students on definitions and other basic facts. Our choice of 100 questions is based on student reactions to longer and shorter lists. If the list seems too long, students won't use it, or not to the level that we want. If the list is too short, the course is (in our view) too easy and too narrow (Kaner, 2003).

We encourage students to refer to the study guide and work through the relevant questions as they take each new section of the class. These help self-regulated learners monitor their progress and understanding—and seek additional help as needed. Students can focus their studying and appraise the depth and quality of their answers before they write a high-stakes exam. Our experience of our students is consistent with Taraban, Rynearson, & Kerr (2000)—many of our students seem not to be very effective readers or studiers, they seem not to be very strategic in the way they spend their study time, and as a result, they don't do as well on exams as we believe they could. We *do not* intend to use this kind of study guide in every course. As we overhaul more courses, we will make strategic decisions as to which courses should offer this form of studying assistance, and which should assume that students have progressed past the need for it.

Given that we use this approach, we gain some benefits (Kaner, 2003):

- This approach gives students notice of the questions and time to prepare thoughtful, well-organized, peer-reviewed answers to them.
- In turn, this allows us to require well-written, thoughtful, well-organized answers on time-limited exams. This maps directly to the testing course's goals of fostering better written communication.
- We can also give students complex questions that require time to carefully read and analyze, but that don't discriminate against students whose first language is not English because they have the questions well in advance and can seek guidance on the meaning of a question from anyone they choose. Additionally, this creates a cooperative learning task involving complex concepts for the whole, in which these students participate—this should help limited-English-proficiency students improve their language skills (Crandall, 1993).
- Students don't have to spend time during the exam to carefully read the question, try to understand it, and then develop an outline for the answer. They did all that during

preparation (or should have). Therefore, the instructor can write an exam with more questions to answer per hour, achieving higher coverage of the course material.

- Using complex questions allows us to insist on precise reading. For example, we often include a question that explicitly states that the student should ignore the possibility of invalid values when generating a set of tests. Those students who create tests for invalid values *lose points*. In the class that we return graded exams and discuss the grading, we draw attention to this error and remind students that testers must be able to read specifications precisely, for example to note what is within the scope of their analysis and what is not. Similarly, when a question asks for three examples, we stop reading after the third, and we say so in the review session. (We tell students these things before the test, but many students seem not to notice it until after the consequence has happened to them or another student they care about.)
- Students can study together, focusing on the structure provided by the study guide. Collaborators review and critique each others' answers, improving the quality of answers and raising the level at which the course content is understood.

We have been surprised by recent published reports that study guides (or "pedagogical aids") either provide no help to students or change their performance by as little as two percent. For example, Dickson, Miller, & Devoley (2005) required students to use the study guide that came with their introductory Psychology textbook, and assessed students' knowledge using a multiple-choice exam. The students who used the study guide scored about 2% higher (a small but statistically significant difference) than students in another section who did not use the study guide. Gurung (2003, 2004) reports even less favorable results of using study guides. Perhaps the difference is that their examinations are multiple choice, while ours are essays that require more demonstration of cognitive structuring of the material. Certainly, other study-assisting activities like concept mapping have significant positive effects on learning. See Zele, Lenaerts, & Wieme (2004) and their many references for more on concept mapping.

Applications to the product under test. The course has a designated software application under test. The Spring 2005 students tested *Open Office's* word processor. Previous semesters focused on the *Open Office* spreadsheet or presentation program, or on the Mozilla *Firefox* browser. Each student joins the open source software project.

Applying what we learn to a sample application has been an important aspect of the testing course. In each of the proposed classes, we look for ways to credibly tie the course material to practical applications. Here are some of the benefits in the testing course.

- It makes the concepts we teach "real" to the students by situating them in the development of products that are well-known and well-regarded (Lave & Wenger, 1991).
- Students gain insight and real-world experience that they can (and do) talk about in job interviews.
- Students create work products that they file with the project (such as bug reports in the project's bug tracking database, or test cases in the collection of test planning materials). They can cite these, show them off and explain them during employment interviews. Feedback from students and industry recruiters tell us several students effectively use these work products in interviews. Knowing that there are stakes outside of the classroom is a strong motivator for some students to do excellent work.
- This approach also facilitates transfer of students' new knowledge and skills to the workplace, because they are doing the same tasks and facing some of the same problems they would face with any commercial software (Clark & Mayer, 2003).

- When students create work products for a project, we can take imperfect examples and rework them in class, step by step, to show what a better job would look like. In the most desirable case, a few students submit work that is blasted as incompetent by an irascible programmer or project manager. That creates a real-world context for the friendly, corrective lecture by the teacher who offers coaching on how to do something like this much better next time. Worked examples can be powerful teaching tools (Clark & Mayer, 2003) especially when they are motivated by real-life situations.

Students work with the product under test in several classroom activities and in their homework assignments.

Classroom activities.

We spend classroom contact hours on coached, small-group activities. For example, groups of up to 4 students apply the lecture material to a real product (*Open Office*) or discuss a hypothetical problem or work on other puzzles, generally problems that the lecture would prepare the student for but that go beyond the lecture itself. We teach in a lab room, with one computer per student. In class, students work together in groups. Activities are open book. Students are encouraged to look up information on-line. The instructor moves from group to group asking or answering questions, giving feedback, or offering supplementary readings that relate to the specific direction taken by an individual group.

Only some of these activities involve the application under test. Others introduce students to the material before the lecture (preparing them for the lecture), develop a theoretical point made in lecture, address a question in the study guide, address a question raised by a student, let them try out a test tool, or help the students work through a complex section of one of the course readings. Students often present results to the class in the last 15 minutes of the 75-minute class. They often hand in their work for grading.

The diversity of activities we imagine is partially inspired by the diversity of examples in the *Activities Handbook(s) for the Teaching of Psychology* (Benjamin & Lowman, 1981; Benjamin, Nodine, Ernst, & Broeker, 1999; Makosky, Sileo, Whittemore, Landry, & Skutley, 1990; Makosky, Whittemore, & Rogers, 1987). We're still learning how to develop good activities and how to document them. Our documentation is free-form so far. Gagnon & Collay (2001) provide what looks like a good structure for thinking about and documenting activities. We intend to experiment with it, and possibly adopt it, over the next year.

Orientation exercises. One interesting type of activity gives the students a problem to work through that they probably won't adequately solve—but that will be solved in the next lecture (Kaner, 2004). "Cognitive conflict or puzzlement is the stimulus for learning and determines the organization and nature of what is learned" (Savery & Duffy, 2001).

Structured discussions. In one of the in-class activities, students get a table. The rows name 16 attributes of good tests (such as power of the test and credibility of the test). The columns list 10 common test techniques. No technique is good on all of the attributes. For example, higher-power tests (tests more likely to expose errors) often use extreme values that are not representative of normal use. The task is to mark the three attributes that a test technique is designed to optimize (domain testing is more focused on the power of tests; scenario testing is usually more focused on realistic uses) and to mark two attributes that testers don't pay attention to when they design a test using this technique. Along with filling out the chart, the student writes a paragraph for each technique, explaining the choices. Students discuss the chart in one class, take it home and think about it, then discuss it again in the next class and submit their work as individuals or as a group. This year, Fall 2005, we will probably spend a third class day on this and ask students to resubmit their analysis close to the end of the course.

This is a difficult but important assignment. It also illustrates the use of an activity to foster a narrowly focused discussion among a group of students.

Drill exercises. In his initial work under NSF Award EIA-0113539 ITR/SY+PE: *Improving the Education of Software Testers,* Kaner expected to be able to bring testing students to mastery of some techniques through practice with a broad set of examples. Padmanabhan (2004) applied this to domain testing in her Master's thesis project at Florida Tech, providing students with 15 classroom hours of instruction, including extensive practice. All students were able to solve the problems in her final examination, which presented problems similar to those solved in class. However, when we added a problem that was just slightly more complex and required application of an idea that had been described in lecture but not practiced, every student failed to discover problems that (in our view) should have been obvious if the student understood the method, rather than followed a rote procedure. This result (and some less formally conducted prior failures) was surprising to us, initially very disappointing, a strong motivator to read further in the STEM education literature and ultimately redesign the testing course.

In retrospect, I realize that we replicated a common finding in mathematics education. Drill helps people master a specific task, but doesn't lead to generalization and transfer (Bransford, Brown, & Cocking, 2000).

We still think that examples and solved problems are very useful, and we still want to present students with strong exemplars, but our goal is to find ways to help students fit them in a broader context, to facilitate effective generalization and successful transfer. Even though students gain hands-on experience with the test techniques (and the other subject matter of the course), the in-class work and the take-home assignments are not drill. Students are not applying a known approach to several similar problems. They are figuring out the details of the approach themselves, applying ideas presented in lecture and comparing notes.

Take-home assignments. We give students about 5 take-home assignments. They have a week or two to complete an assignment. It often comes with a grading rubric. Eventually, all of them will come with rubrics and some will come with samples of answers to similar assignments. A typical assignment applies test techniques to the product under test, gathering information about the product, its markets and/or its risks, working through a standard or trying out a tool and writing an experience report. Students are welcome to work together in groups of two or three.

Notes on Video Implementation

We are amateurs at photography and video production, not particularly talented, but enthusiastic. Our videos reflect that. Here are some good examples of our work.

- This one (http://www.testingeducation.org/k04/video/FunctionTest.wmv) on function testing is a self-contained video that includes a detailed demonstration of the technique.
- These two (http://www.testingeducation.org/k04/video/BBSTguiAuto2.wmv and http://www.testingeducation.org/k04/video/OverviewPartB.wmv) illustrate our current layout and pacing.

We've made progress in laying slides out to be more readable, in editing the video to avoid jarring transitions, and in pacing the lecture and the slide transitions, but we have a long way to go. We expect to continue to make incremental progress (remaining obvious amateurs) in technical production, but we hope to make the content more engaging in a few ways. First, we expect to provide more demonstrations, like the function testing one. Second, we want to add more engaging stories, like http://www.testingeducation.org/k04/video/BBSTscriptTest2.wmv. Unfortunately, these take a lot of work. Third, we're intrigued by infomercials, which often heighten interest through interactions among several actors, including scripted discussions that pose questions to address specific, predictable areas of confusion or resistance to a new idea.

We record the videos at home using a mid-priced digital camcorder (a Sony DCR TVR-38). We download the videos to a Dell XPS Gen5 computer with a dual-core Pentium chip with hyperthreading.

Processor speed and the ability to do multiprocessing are important for editing and rendering (saving the edited video in a final format, in our case QuickTime or Windows Media Video). A video that renders in an hour (on the Gen5) rendered in up to 8 hours on slower machines. When an error, such as corrupted input, a memory leak or other bug, halts the rendering process, rendering seems to either fail right away or near the end of processing. On the predecessor to the Gen5 (an Alienware Roswell with a Matrox video coprocessor), failures were common—redoing the rendering could add 30 hours of rework for a 45-minute lecture segment. Producing videos on schedule, on time for students to be able to watch them before the scheduled class, is much more challenging the slower the rendering process and the less reliable the computer. We use the Windows systems in order to use *Adobe Premiere Pro*. Both of us have our own Macintoshes and Windows systems (and have for years). We prefer the Macs, but our pilot studies (two full lectures and some other fooling around) convinced us that *Premiere* was more reliable—at least in the way we worked with video editing—than *Final Cut Pro*, and *Premiere* isn't being updated for the Mac. Looking at current prices for our setup, it costs about $8000 for a fast and reliable computer with lots of memory, three hard disks (source data, resulting video, other data and work area), *Premiere*, and some utilities like *SnagIt* (for screen shots and videos of running programs).

Creating a lecture includes preparation tasks as well as taping and editing a video. These include polishing the slides to get the flow of the lecture right, (very imperfectly) balance the time to be spent per slide, laying out the slides to be readable after video rendering in a compressed format, writing out detailed notes for the lecture and rehearsing the lecture before taping.

Giving the lecture, on tape, also takes time. We might tape the presentation of a single slide several times until it finally feels right, then move to the next slide. An hour of final tape might take 3 to 10 hours of videotaping, followed by 4.5 to 15 hours (about 1.5 hours per hour of tape) of editing.

From "start" to finish, where "start" means we already have good slides and a fairly clear idea of the lecture (with lecture experience with these slides), lecture material to support one classroom contact hour (perhaps 30-50 minutes of video) took about 35 hours last year (the unreliable and slightly slower Alienware system). On the newer system, we hope to stabilize at 25 hours video preparation time per class.

Along with video preparation is preparation of all of the other supporting materials. Freeing up all of the classroom contact hours for labs and seminars means that we have to create laboratory and seminar activities, develop ways to evaluate them, and provide feedback to the students.

The overall preparation time cost, borne primarily by the faculty member, is enormous and far beyond what we suspect any university would pay for. It is a labor of love. For us, the driving passion is for improvement of the state of the practice in the broad field of software testing. In a field that is educationally underserved (academically and commercially), providing high-quality free course materials to the community at large might have an impact on the development of basic knowledge and skill in the field.

To enable this broad distribution, we have to keep the intellectual property rights to our course. That makes it possible for us to license the material any way we want, including Creative Commons. Some schools (K-12 and university) have begun asserting ownership of copyrightable teaching materials created by their faculty. In these cases, the school might not approve of publishing courseware that they own, for free.

We planned for this by setting up to do all of the work at home, using our own equipment, and so far, we have done all of the videotaping and editing at home. Eventually, Florida Institute of Technology approved a stored course policy that allows us to keep ownership of materials we create at school so long as we don't rely heavily on school resources (The final policy is almost the same as the proposal we reported in Kaner, 2002). This has allowed us to supplement our materials with work done by research

assistants in Kaner's lab at Florida Tech and will allow us to do more work at school and more in collaboration with other faculty, in the future.

A Broader Market

The course appears to be working well in our classes at Florida Tech. We are still evaluating comparable performances (such as final exams from previous teachings of the course, compared to those from the new approach). Our impression from informal comparison is that the students are generating better work. We are also getting favorable student evaluations.

In addition to local teaching, we make the course available on the Web, for free. About 120 people have signed up for a low-traffic, moderated email list for people who teach a testing course using our materials. Some teach it at university, others at their own company. Some are self-studiers trying to gain insight from discussions among the teachers. The list has been in operation for about six months, but we are still just getting started on serious discussions. Eventually, we expect feedback from this list and from a second list (discussion by students) to provide a steady stream of insights into the weaknesses of the course.

Informal feedback so far is positive and enthusiastic, but it is clear that these materials are far from ideal for self-study and that several instructors (especially commercial trainers) could use more help on evaluating the results of tests and assignments.

Closing Notes

This has been a time-consuming course to create, and we're not done. We have more lectures to tape, plus more grading samples, examples, and other instructional supports.

We see the work as worth the effort, partially because it appears to be improving our on-campus students' learning and enthusiasm, but especially because it is being reused around the world and will be used even more broadly as it gets more polished. We also expect the availability of a good free course to stimulate improvement in the leading commercial courses, in order to compete well with our free offering. Some commercial instructors are already incorporating some of our materials in their courses (which they are welcome to do under our Creative Commons license). Others, we are told, have started their upgrading process without using our material.

We see this work as promising enough that we are already planning the next few courses. Eventually, we hope to offer an open source curriculum in software testing, rather than just one or two courses. As software development work disperses ever more around the world, a set of high quality courses available to everyone for free offers the possibility of a reasonably high baseline level of knowledge of the field everywhere—or as competition develops, of several good baselines that present competing visions of the field and its operations.

References

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. A., Pintrich, P. R., et al. (2001). *A Taxonomy for Learning, Teaching & Assessing: A Revision of Bloom's Taxonomy of Educational Objectives* (Complete Edition ed.). New York: Longman.

Angelo, T. A., & Cross, P. K. (1993). Teaching Goals Inventory. Retrieved October 10, 2005, 2005, from http://www.uiowa.edu/~centeach/tgi/index.html

Benjamin, L. T., & Lowman, K., D. (Eds.). (1981). *Activities Handbook for the Teaching of Psychology* (Vol. 1). Washington, DC: American Psychological Association.

Benjamin, L. T., Nodine, B. F., Ernst, R. M., & Broeker, C. B. (Eds.). (1999). *Activities Handbook for the Teaching of Psychology* (Vol. 4). Washington, DC: American Psychological Association.

Bligh, D. A. (2000). *What's the Use of Lectures?* (American ed.) San Francisco: Jossey-Bass.

Bloom, B. S. (Ed.). (1956). *Taxonomy of Educational Objectives: Book 1 Cognitive Domain*. New York: Longman.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.). (2000). *How People Learn: Brain, Mind, Experience and School (Expanded Edition)*. Washington, D.C.: National Academy Press.

Brooks, L. R. (1978). Non-analytic concept formation and memory for instances. In E. Rosch & B. B. Lloyd (Eds.), *Cognition and categorization* (pp. 169-211). Hillsdale, NJ: Erlbaum.

Burnstein, I. (2003). *Practical Software Testing*. New York: Springer.

Clark, R. C., & Mayer, R. E. (2003). *e-Learning and the Science of Instruction*. San Francisco, CA: Jossey-Bass/Pfeiffer.

Crandall, J. (1993). Content-Centered Learning in the United States. *Annual Review of Applied Linguistics, 13*, 111-126.

Dannenberg, R. P., (n.d.). Just-In-Time Lectures. Retrieved March 12, 2005 from http://www.jitl.cs.cmu.edu/jitrbd.htm.

Dickson, K. L., Miller, M. D., & Devoley, M. S. (2005). Effect of Textbook Study Guides on Student Performance in Introductory Psychology. *Teaching of Psychology, 32*(1), 34-39.

Felder, R. M., & Brent, R. (1996). Navigating the bumpy road to student-centered instruction. *College Teaching, 44*, 43-47.

Fintan, C. (2000). *Lecturelets: web based Java enabled lectures.* Paper presented at the Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland.

Firstman, A. (1983). A comparison of traditional and television lectures as a means of instruction in biology at a community college.: ERIC.

Ford, D. G. (2002). Teaching anecdotally. *College Teaching, 50*(3), 114-115.

Forsyth, D., R. (2003). *The Professor's Guide to Teaching: Psychological Principles and Practices*. Washington, D.C.: American Psychological Association.

Gagnon, G. W., & Collay, M. (2001). *Designing for Learning: Six Elements in Constructivist Classrooms*. Thousand Oaks, CA: Corwin Press.

Gurung, R. A. R. (2003). Pedagogical Aids and Student Performance. *Teaching of Psychology, 30*(2), 92-95.

Gurung, R. A. R. (2004). Pedagogical Aids: Learning Enhancers or Dangerous Detours? *Teaching of Psychology, 31*(3), 164-166.

Hamer, L. (1999). A folkloristic approach to understanding teachers as storytellers. *International Journal of Qualitative Studies in Education, 12*(4), 363-380.

He, L., Gupta, A., White, S. A., & Grudin, J. (1998). *Corporate Deployment of On-demand Video: Usage, Benefits, and Lessons* (No. MSR-TR-98-62). Redmond, WA: Microsoft Research.

IEEE. (1983). *IEEE Standard 829-1983 (Reaffirmed 1991) Standard for Software Test Documentation* Piscataway, NJ: IEEE Standards Dept..

IEEE. (1990). Std 610.12-1990, Standard glossary of software engineering terminology. Piscataway, NJ: IEEE Standards Dept.

Kaner, C. (1988). *Testing Computer Software* (1st ed.). New York: McGraw Hill.

Kaner, C. (2001). *Measurement issues and software testing (Keynote address)*. Paper presented at the QUEST 2001 Segue Software User's Conference. Retrieved April, 2005. from http://www.kaner.com/pdfs/measurement_segue.pdf.

Kaner, C. (2002). The proposed Florida Tech stored course policy. *Computer Graphics, 36*(2), 15-17, 21-22.

Kaner, C. (2003, February). *Assessment in the Software Testing Course.* Paper presented at the Workshop on the Teaching of Software Testing (WTST), Melbourne, FL.

Kaner, C. (2004, February). *Carts before horses: Using preparatory exercises to motivate lecture material.* Paper presented at the Workshop on Teaching Software Testing, Melbourne, FL.

Kaner, C., Bach, J., & Pettichord, B. (2001). *Lessons Learned in Software Testing*: Wiley.

Kaner, C., & Bond, W. P. (2004). *Software engineering metrics: What do they measure and how do we know?* Paper presented at the 10th International Software Metrics Symposium (METRICS 2004). Retrieved September, 2005 from http://swmetrics.mockus.us/metrics2004/lbp/KanerBond.pdf.

Kaner, C., Falk, J., & Nguyen, H. Q. (1993). *Testing Computer Software* (2nd ed.): International Thomson Computer Press.

Kaner, C., Falk, J., & Nguyen, H. Q. (1999). *Testing Computer Software* (republished 2 ed.). New York: John Wiley & Sons.

Kaufman, J. C., & Bristol, A. S. (2001). When Allport met Freud: Using anecdotes in the teaching of Psychology. *Teaching of Psychology, 28*(1), 44-46.

Langer, N. (2002). Enhancing adult learning in aging studies. *Educational Gerontology, 28*(10), 895-904.

Lave, J., & Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge, England: Cambridge University Press.

Maki, W. S., & Maki, R. H. (2002). Multimedia comprehension skill predicts differential outcomes of web-based and lecture courses. *Journal of Experimental Psychology: Applied, 8*(2), 85-98.

Makosky, V. P., Sileo, C. C., Whittemore, L. G., Landry, C. P., & Skutley, M. L. (Eds.). (1990). *Activities Handbook for the Teaching of Psychology* (Vol. 3). Washington, DC: American Psychological Association.

Makosky, V. P., Whittemore, L. G., & Rogers, A. M. (Eds.). (1987). *Activities Handbook for the Teaching of Psychology* (Vol. 2). Washington, DC: American Psychological Association.

Marick, B. (1997). *Classic testing mistakes*. Paper presented at the Software Testing Analysis & Review Conference (STAR 97). Retrieved October 10, 2005 from http://www.testing.com/writings/classic/mistakes.html.

Medin, D., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review, 85*(207-238).

Myers, G. J. (1979). *The Art of Software Testing*. New York: Wiley.

Padmanabhan, S. (2004). *Domain Testing: Divide & Conquer.* Unpublished M.Sc. Thesis, Florida Institute of Technology, Melbourne, FL.

Rossman, M. H. (1999). Successful online teaching using an asynchronous learner discussion forum. *Journal of Asynchronous Learning Networks, 3*(2).

Saba, F. (2003). Distance education theory, methodology, and epistemology: A pragmatic paradigm. In M. G. Moore & W. G. Anderson (Eds.), *Handbook of Distance Education* (pp. 3-20). Mahwah, New Jersey: Lawrence Erlbaum Associates.

Savery, J. R., & Duffy, T. M. (2001). *Problem Based Learning: An Instructional Model and Its Constructivist Framework* (No. CRLT Technical Report No. 16-01). Bloomington, IN: Indiana University.

Smith, D. J. (2005). Wanted: A New Psychology of Exemplars. *Canadian Journal of Psychology, 59*(1), 47-55.

Taraban, R., Rynearson, K., & Kerr, M. (2000). College students' academic performance and self-reports of comprehension strategy use. *Reading Psychology, 21*(4), 283-308.

Whittaker, J. (2002). *How to Break Software*. Boston: Addison-Wesley.

Williams, R. G., & Ware, J. E. (1977). An extended visit with Dr. Fox: Validity of student satisfaction with instruction ratings after repeated exposures to a lecturer. *American Educational Research Journal, 14*(4), 449-457.

Zele, E., Lenaerts, J., & Wieme, W. (2004). Improving the usefulness of concept maps as a research tool for science education. *International Journal of Science Education, 26*(9), 1043-1064.

Cem Kaner (Author), Rebecca L Fiedler (Author). 5.0 out of 5 stars 4 ratings. See all 5 formats and editions Hide other formats and editions. Price.Â Cem Kaner, J.D., Ph.D., is Professor of Software Engineering at Florida Institute of Technology, a research-focused university that was recently ranked as one of the top 200 universities in the world. Kaner is lead author of several books, including Lessons Learned in Software Testing, The Domain Testing Workbook, and Testing Computer Software.Â Kaner holds doctorates in Psychology and in Law. He teaches courses on software testing, metrics, requirements analysis, applied statistics, and software-related law, ethics and societal issues. It shows two interviews made a year after two couples met in unusual circumstances, similar to those described in the Studentâ€™s Book. British Versions. Student's Worksheets 0.84mb pdf.Â Jeremy Sharples guides the students round a dayâ€™s walking tour of part of central London. He starts in Leicester Square and goes on to Convent Garden market to do some shopping. He then watches the street entertainers in Covent Garden Square and visits some designer shops. What do computer scientists do? What does it take to make a computer or a computer program work? We answer these questions and more with MyCS: Computer Science for Beginners. We believe that anyone can succeed in and enjoy computer science. This course is an early introduction to CS, designed for anyone who's completely new to the field. It explores a combination of the basic principles of how computers work and how we can use them to solve interesting problems and create amazing things. Lessons alternate between general exercises and assignments in Scratch, which offer a chance to both p