Building a Free Courseware Community Around an Online Software Testing Curriculum

Cem Kaner   kaner@kaner.com
Professor of Software Engineering
Florida Institute of Technology

Rebecca Fiedler   fiedler@beckyfiedler.com
Assistant Professor of Education
Indiana State University

Scott Barber   sbarber@perftestplus.com
Executive Director
Association for Software Testing

Key Points

1. Potential for developing / sustaining reusable professional development courseware via a free software model

2. Reusing learning objects from the PD courses highlights potential for synergy between academic and professional development courses

Key Challenges

- Training model

- Adoption model

- Development model

- Recruitment model

- Funding infrastructure

- Instructor training model

- Business model

- Maintenance model

- Instructor retention

- Intellectual property

Building a Free Courseware Community Around an Online Software Testing Curriculum

Eighth Annual MERLOT International Conference, Minneapolis, MN, August 9, 2008

CEM KANER

Professor of Software Engineering, Florida Institute of Technology. kaner@kaner.com

REBECCA L. FIEDLER

Assistant Professor of Education, Indiana State University. fiedler@beckyfiedler.com

SCOTT BARBER

Executive Director, Association for Software Testing. sbarber@perftestplus.com

ABSTRACT

We are adapting a rich collection of Creative Commons academic course materials (video lectures, slides, assessment materials) in a course on Black Box Software Testing (BBST) for a professional audience (Association for Software Testing [AST]). AST's model breaks BBST into learning units, each presented as a 4 week, online course. Students spend 8-12 hours per week on lectures, readings, quizzes, individual and group assignments, an essay exam, and extensive peer review. AST offers these courses to its members for free. The typical class has 16 students from 3 or 4 continents. This presentation overviews the BBST materials and the AST adaptation, then describes the model under which we recruit, train, and keep AST instructors involved as volunteers teaching and maintaining the course.

INTRODUCTION: AST'S BLACK BOX SOFTWARE TESTING (BBST) COURSES

In a recent survey of software development managers, the most often cited top-of-mind issue was software testing and quality assurance (Zeichick, 2005). Testing is not quality assurance--a brilliantly tested product that was poorly designed and programmed will end up a well-tested, bad product. However, testing has long been one of the core technical activities that can be used to improve quality of software. Many organizations invest heavily in testing. For example, most Microsoft projects employ one tester per programmer (Cusumano & Selby, 1998; Gates, 1997).

Despite enormous investment in testing *work*, testing *practice* is insufficient. For example, *The Economic Impacts of Inadequate Infrastructure for Software Testing* (National Institute for Science & Technology, 2002) estimates that software weaknesses cost the U.S. economy $59 billion per year. This is a lower bound estimate. NIST's calculations miss (among others) the recurring costs of dealing with viruses and worms that exploit holes in current software (estimated as high as $45 billion worldwide for 2002 and $119 to 145 billion for 2003) (Jenkins, 2003) and the many "one-time" software crises, such as $300-600 billion spent in 1996-2000 to fix the Year 2000 bug (Leuning, 1999), the loss of NASA's Mars Climate Orbiter (Oberg, 1999), and a defect that crippled the USS Yorktown for almost 3 hours while at sea (Slabodkin, 1998). Better testing would not eliminate these costs, but weak testing contributes to them.

The need for skilled testing is greater, not less, when companies and government agencies outsource software development. If a company can't control how a product is made, it must carefully check what it gets. To deal efficiently with contracted software, we must improve our ability (including our educational support for developing that ability) in rapid, risk-focused investigation of contractors' products (National Defense Industrial Association, 2006).

Despite its significance in the workforce, testing is only lightly covered in university courses and superficially covered in most commercial courses. As individuals and as a professional society (the Association for Software Testing) that caters more to experienced practitioners than newcomers, we have been developing alternative courseware and teaching models in an effort to improve the effectiveness and broaden the scope of software testing education.

As our projects have evolved, we have been developing academic and professional development materials in parallel, often passing things between courses. To keep the scope of this report manageable, we will narrow our focus to the Association for Software Testing's courses on black box software testing (BBST). ("Black box" testing involves treating the program as a black box, whose internal structure cannot be seen. Tests often challenge the acceptability of the software to the intended user, the safety, security and responsiveness of the system that contains this software, or the ability of the software to perform properly in diverse environments or properly interact with other subsystems.)

BACKGROUND OF THE AST COURSE MATERIALS

The AST courses are based on an introductory course in software testing (the Black Box Software Testing Course, BBST) that evolved over a period of 25 years, from informal in-house training to highly successful commercial professional development (PD) to a multimedia hybrid for undergraduates and graduate students and back to online PD.

As a commercial course, BBST spun off *Testing Computer Software* (1st edition, Kaner, 1988; 2nd edition, Kaner, Falk, & Nguyen, 1993), *Bad Software: What To Do When Software Fails* (Kaner & Pels, 1998) and *Lessons Learned in Software Testing* (Kaner, Bach, & Pettichord, 2001) and sets of course notes of up to 1000 slides. Kaner co-developed and co-taught the course with several leading practitioners, including Hung Quoc Nguyen, Doug Hoffman, Elisabeth Hendrickson, and James Bach who contributed so much from his courses that he appears as co-author on current slide sets.

By 2000, when Kaner joined Florida Institute of Technology, this was a polished, financially successful course that had gone through extensive peer review, hundreds of teachings and conference presentations, about a dozen major revisions, and several dozen smaller, but significant, ones.

At Florida Tech, Kaner began adapting this course to an academic audience. The National Science Foundation contributed substantially to the adaptation, as did Texas Instruments. Rational Software (now IBM) funded a re-adaptation of the academic course to the practitioner market, creating Rational's introductory testing course, *Principles of Software Testing for Testers* (Kaner, 2002).

In 2004, Kaner began videotaping lectures. Many of the students' most powerful learning experiences came in informal weekend meetings, preparing assignments and exams. To make it possible to bring this type of work into the classroom, we decided to move lectures out of the classroom, taping lectures and requiring students to watch them before coming to class.

Live teachings of BBST had been videotaped before (Kaner, 1995, 2002), but the new videos were intentionally designed for web-based study rather than taping live classroom presentation. Kaner lectured in a quiet room: an hour of final video was edited down from 3 to 10 hours of taped lecture. This allowed us to move lectures out of the classroom. Students watched lectures from a website before coming to class and then participated in discussions or coached activities in class. We found that this approach (combined with 17 other instructional choices) was popular with our students and that they worked unusually hard in the course (Kaner, 2008; Kaner & Fiedler, 2005a, 2005b, 2007a). Other instructors have also had success with hybrid courses that place video lectures out of the classroom (Day & Foley, 2005; J. A. Day & J. Foley, 2006; Day, Foley, Groeneweg, & Van Der Mast, 2004, 2005; J. A. Day & J. D. Foley, 2006; Foley & Day, 2004-2006; Gannod, Burge, & Helmick, 2008).

The existence of the video lectures laid a foundation for a purely web-based course.

We started by posting the lectures at our lab's website, www.testingeducation.org/BBST and opening a discussion group on Yahoo Groups. Lots of copies of videos were downloaded but there was little discussion. Our impression from email and a few face-to-face conversations was that people were watching only the first few videos. Therefore, we decided to create an instructor-led course.

Starting in October, 2006, we experimented with online variations of a course that included all of the material offered in the 1-semester Florida Tech academic course. This culminated in an Instructors Course that started in January 2007 and collapsed in March 2007.

One of the key problems for the 2007 Instructors Course was that testing is a demand-driven activity: development groups want their products tested as soon as they are ready for testing, and testing schedules are often compressed because the development group works beyond its scheduled delivery date. As a result, at times outside their control, testers hit an on-the-job crisis and can no longer participate in a course. Our impression from the 2007 group, which was strongly reaffirmed in a face-to-face Instructor Course in July 2008, we concluded that testers could usually see about a month ahead, but not beyond that. Accordingly, we decided to split BBST into a series of mini-courses that lasted 1 month each.

LEARNING OBJECTIVES OF THE AST COURSES

> AST views software testing as an empirical, technical investigation conducted to provide stakeholders with quality-related information... AST views software testing as a cognitively complex activity, requiring critical thinking, effective communication, and rapid self-directed learning. (Association for Software Testing, 2007b)

Software testing is often confused with manufacturing quality control (QC). Manufacturing QC looks for manufacturing errors: there is a clear standard for a "good widget" and deviations from it are defects. QC tasks involve routine inspection of each widget (or a sample) for each way that the specific widget could deviate from the "good widget."

Software errors are not manufacturing errors, defects that appear in some copies of the product but not others. Software errors are design errors. They appear in all instances of the product. This creates several cognitively complex challenges, such as:

- ▪ *The impossibility of complete testing* (Kaner, 1997; Myers, 1979). Two tests are distinct if a program can pass one test while failing the other. The number of distinct tests of any nontrivial program is impossibly large and so software testing becomes a challenging sampling problem. Which tiny subset of the tests we *could* run will probably yield the most information today?

- ▪ *The oracle problem* (Bach, 1999; Hoffman, 1998, 1999). In theory, an oracle provides a mechanism for determining whether the program's behavior was correct, given a set of preconditions, a specified action, and the observed results. However, we never fully specify the preconditions or the observable results. For example, it is rare to specify how much free memory is available at the start of a test or how fragmented it is, or how hot the central processor chip is, but each of these preconditions has been critical for some product failures. Similarly, we can specify that 2+3 should yield the expected result of 5, but do we check for memory leaks or for unexpected side effects (such as disconnecting the system from the network)? The tester has to decide what aspects of the test to pay attention to and how to troubleshoot hard-to-reproduce failures (failures that involve conditions that the tester didn't realize s/he should attend to).

- ▪ *The testing mission and objectives*. In an influential paper, Marick (1997) distinguished between two very different testing objectives: hunting for bugs versus helping the project manager make a release decision. He showed that the testing strategy for good bug hunting differed strongly from the strategy for good status assessment. This lesson extends across many other objectives. Testers are often asked to assess the conformity of a product with its specifications, help predict and control the costs of product support, help minimize the risk of safety-related lawsuits, check interoperability

with other products, or evaluate a product for a third party (such as a company considering buying the company that made of this program, or an attorney considering suing the manufacturer). Thus, a critical challenges of testing is deciding what objectives are appropriate to the project, and then deciding which test techniques will help the testers achieve those objectives.

- ▪ *Distributed teams.* On logistically simple projects, programmers do some of the testing and testers who work in the same location do the rest of the testing. On more complex projects, parts of the testing are outsourced and managing the overall testing project involves coordinating work across separate testing teams working on several different continents.

Challenges like these make testing cognitively complex. Therefore, rather than focusing on technology or on one-size-doesn't-really-fit processes, we are building the AST courses around the development and application of several cognitive skills to the tasks of testing.

These are the primary objectives of the BBST course series:

- ▪ 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);

- ▪ Improve students' study skills, reading skills and test-taking skills--these are important for handling increasingly difficult classroom assignments and also for competent testing (such as deriving test cases from specifications);

- ▪ Gain (and document) experiences that can improve the student's chances of getting a job in testing.

Individual AST courses will address some of these objectives but not others, and will also add objectives specific to the content of the course.

The underlying intention of the AST courses is to *improve the state of the practice* and for that, we believe that we must foster higher-level knowledge (Anderson, et al., 2001), developing skill and insight. We are specifically *not* interested in driving students to adopt a common vocabulary or in memorization of basic terms and concepts. Some of that will happen in the normal course of things, but these are not our learning objectives.

STRUCTURE OF THE AST COURSES

AST courses are fully online. They are free to all members. Classes are capped at 20 students and the typical class has students from several continents. Currently, all instruction is in English.

We have developed two AST courses so far:

- ▪ *BBST Foundations* is a prerequisite for all subsequent classes. It introduces a common vocabulary and introduces four key complexities in software testing: the need to tailor the mission of the testing effort to the project's circumstances and the stakeholders' goals; the heuristic nature of test oracles; the difficulties of measurement and risks of dysfunctional measurement (Austin, 1996); and the impossibility of complete testing. Students complete a group assignment (instructors select groups to maximize geographic and cultural diversity), do homework, take intentionally-difficult tests and an exam, and peer review each other's work. Most students finished school long ago and many have emotional reactions to a course with formal assessments. Along with teaching content, we are

teaching students how to work well in online courses and how to cope with assessment. For a full list of learning objectives, see (Association for Software Testing, 2007a).

- **BBST Bug Advocacy** (Association for Software Testing, 2008a) presents bug reporting as a persuasive communication task with the goal of providing stakeholders with well-written reports that carry solid technical information and facilitate wise decisions about which bugs to fix. The course also introduces testers to a standard application (Open Office) that we will test in all subsequent BBST courses and to the Open Office bug reporting system. It also drives students through a much more complex, multi-phase group project that demands better peer review. This is a prerequisite for all AST courses (other than Foundations).

We have video lectures (at http://www.testingeducation.org/BBST/) to support another 11 courses, slides and other materials to support at least 4 additional courses and expect to release several new AST courses based on these over the next year.

Figure 1 presents the structure of the AST-BBST course. All BBST courses are variations on this theme.

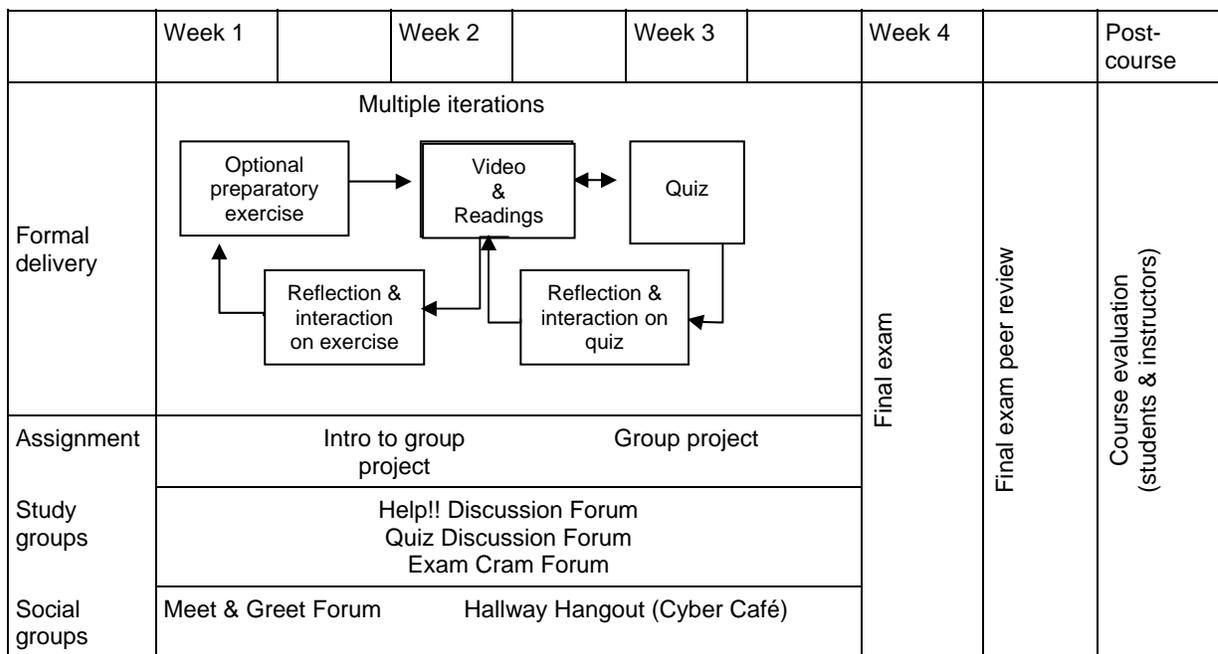|	Week 1		Week 2		Week 3		Week 4		Post-course
Formal delivery	Multiple iterations: Optional preparatory exercise → Video & Readings ↔ Quiz; Reflection & interaction on exercise; Reflection & interaction on quiz						Final exam	Final exam peer review	Course evaluation (students & instructors)
Assignment		Intro to group project			Group project				
Study groups		Help!! Discussion Forum / Quiz Discussion Forum / Exam Cram Forum							
Social groups	Meet & Greet Forum		Hallway Hangout (Cyber Café)						

Figure 1: The AST-BBST COURSE MODEL

Throughout the course, we strive to incorporate both the formal and informal aspects of face-to-face instruction, including both instructor-focused tasks as well as social interactions. To preserve the focus of task-oriented discussions, we create separate space for purely social discussion.

- Each AST course runs four weeks, Sunday to Saturday. Each week is split into two segments:
  - the first segment runs Sunday to Wednesday
  - the second segment runs Thursday to Saturday.
  - Segments end at midnight local time: we enforce deadlines at 8 am GMT.

Thus, each course has eight segments.

- **Meet & Greet:** Each course starts with a Meet & Greet activity, where students get to know who else is in the class.

- ***Hallway Hangout:*** Along with *Meet & Greet,* we encourage students to socialize in the "Hallway Hangout" forum. (By "forum", we mean a named threaded-discussion area that allows students to create topics and respond to posts from other students.)

- ***Study Guide and Exam Cram Forum****.* We create the closed-book final exam from a set of essay questions published in a study guide at the start of class. We encourage students to refer to the study guide and work through the relevant questions as they take each segment of the class. They prepare for the exam by drafting answers to the questions, posting and critiquing each others' drafts in the *Exam Cram Forum.*

Our study guides have about 20 questions, half for short-essay (requiring a paragraph or two) and half for long-essay (requiring about three times as much as the short essay). Our choice of 20 questions reflects our experience with academic students that 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, it is (in our view) too easy and too narrow (Kaner, 2003).

This approach creates a cooperative learning task involving complex concepts--this should help limited-English-proficiency students cope with the course and incidentally improve their language skills (Crandall, 1993). Additionally, because the time limits (for studying and for the exam) are generous, speed pressures don't unfairly disadvantage students who are naturally slower because they are less fluent in English.

The main benefit of this approach, as we see it, is that, given lots of time for preparation time and opportunity for peer-review, we can require more carefully considered, better written answers. We can punish shotgun answers, irrelevant answers, answers that assume away the question's difficult parts, and disorganized answers. We explain our grading standards to students, even providing a video lecture in which I grade four answers (composites of genuine student answers, ranging from excellent to terrible) and explain my reactions to the answers as I go (Kaner & Fiedler, 2007b).

Our approach helps 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. However, in practice, few students take advantage of this in BBST Foundations and most have trouble with the final exam. This is consistent with Taraban, Rynearson, & Kerr (2000)--many of our students (undergraduates, graduates, and practitioners) 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.

More students use the *Exam Cram Forum* in our second course, BBST-Bug Advocacy, and we expect higher rates in later courses. This is consistent with the patterns we see in the academic version of BBST, which has two mid-terms and a final. Few students prepare answers to study guide questions for the first midterm and their results are poor. Some prepare more thoroughly and do better on the second midterm and the final.

We have been surprised by 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.

- ▪ *Videotaped lectures.* The course lecture includes 1 to 4 hours of videotaped material. We split the lectures into separate files and tell students which files to watch in which segment (for example, watch lecture 1 by the end of the first half-week (first segment) of the course).

To this point (August, 2008), all lectures are by Cem Kaner. Over the next 12 months, we expect to add short clips (1 to 10 minutes) by other instructors that provide an example, rebut a viewpoint presented in the main lecture, explore a related topic, or otherwise supplement the lecture. This will improve the diversity (of viewpoints as well as role models) of the material and will serve as a training ground for new lecturers.

According to Bligh's (Bligh, 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."

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.)

Stored lectures offer several benefits over live:
- 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--jump backwards in the video to review what was said (He, Gupta, White, & Grudin, 1998).
- Students can replay the lecture while studying for an exam or working on an assignment, reviewing the material in the context of a specific problem or task.
- Students whose first language is not English often have difficulty keeping up with live lectures and can 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 no interruptions.
- Live lecturers sometimes drift or ramble. Stories are not always directly relevant and some take so long that students lose the message. We script lectures (using a teleprompter) and allow spontaneous departure from the script during taping for

spontaneous remarks. We can cut or shorten an example or digression that takes too long or weakens the focus during post-production.

However, live lectures *do* (or at least can) ***provide*** opportunities for interaction between the lecturer and the student. Taped lectures do not. To compensate for this, we train AST instructors to be visibly present in the online course (Fiedler, 2008a, 2008b, 2008c). Being present doesn't mean being dominant. For example, a teacher can show s/he is paying attention by saying something as simple as "thank you" in response to a submission, without commenting, at that time, on the content at all. We generally let students wrestle with the content of a discussion for a few days before giving our opinions.

In BBST-Foundations, students watch video lectures only in the first four segments. Segments 5 and 6 are reserved for the group project and preparation for the final exam. In BBST-Bug Advocacy, students watch video lectures in all of the first six segments. We intend to limit lectures in future courses to the first four or (at most) five segments.

▪ *Preparatory exercises.* We use preparatory exercises to introduce students to key ideas in the lectures. Before watching the lecture, students work through the exercise, posting their answer to a series of questions on a discussion forum. We encourage students to time-limit their own work to 30 minutes, then review each other's answers, then watch the lecture. We don't expect students to solve the exercise. We expect them to gain an appreciation of the difficulty of the problem--which we then solve in the lecture (Kaner, 2004). Working on a discovery activity before the lecture improves learning (Schwartz & Bransford, 1998). "Cognitive conflict or puzzlement is the stimulus for learning and determines the organization and nature of what is learned" (Savery & Duffy, 2001). After they watch the lecture, we ask students to review their answer and the answers of one or two students, reflecting on what they learned about the problem.

We provide two examples of our preparatory exercises in Appendices A and B.

The diversity of our exercise and assignment structures 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) and Sattsangi, Rutledge, Cooper & Fox (2008; Science Education Resource Center, undated) provide what looks like a good structures for thinking about and documenting activities. We intend to experiment with these over the next year.

▪ *Quizzes*. We provide a quiz with objective questions (typically multiple choice) for almost every lecture. The questions are difficult. They require precise reading. Our standards are at (Kaner, 2007). Students take these quizzes *open book*. We encourage them to answer quiz questions while they watch the lecture. The quiz helps the student understand whether s/he understands the basics of what is being presented. We run our courses under Moodle, and Moodle presents the quizzes. When a student answers a question incorrectly, Moodle presents feedback that we wrote for that particular incorrect answer. (We have this level of feedback for most, but not all questions.)

▪ *Quiz Discussion Forum*. After the quiz, we post every question, with its answer, as a separate thread in the discussion forum. Students discuss the questions, challenging the answer or justifying an answer they prefer. In BBST-Foundations, students often fare poorly on the first quizzes. The typical student has been away from school for years, and hasn't had to deal with well-written, difficult multiple choice questions. Many students respond emotionally when they fail the first quiz. It brings back bad memories of badly-written, unfairly graded exams from years past. The forum gives students a chance to blow off steam and develop bonds with other students. After a few days of student comments, the instructor responds, explaining the rationale of the question and answer,

sometimes agreeing with a student's analysis and describing how the question will be improved for the next class.

In the AST courses, quizzes are low-stake assessments. They play a negligible role in the final evaluation of the student's work. More than any other part of the course, the discussions of quizzes in BBST-Foundations convey to the students that every assessment activity in our courses is primarily a learning activity, secondarily a way for the student to assess their own knowledge, and then to varying degrees, a summative evaluation tool for the instructor.

Discussions in the BBST-Foundations quiz forum are often difficult, but they build trust. Discussions in BBST-Bug Advocacy are much less intense, probably because everyone in Bug Advocacy has successfully completed Foundations. They (usually) understand the AST course structure and approach and trust the instructor.

- ▪ *Help Forum.* This is where students ask for any type of course-related help, such as questions about technology, requests for supplementary readings, and gripes about the content of the lecture.

- ▪ *Assignments.* Each AST course has one or two assignments.

We intentionally maximize diversity of the students. In the early courses, students are still learning how to collaborate across time zones (a typical student team has a student from Asia or Australia, a student from Europe or Africa or the Middle East, a student from the east coast of the United States or eastern South America, and a student from the western United States or western South America), differences in standard approaches to problems based on different cultures and industrial backgrounds, etc. For the first AST courses, we are sticking with one assignment. Later, as students become more efficient global collaborators, we will often have two assignments.

The prototypic assignment applies lessons of the course to an open-source program in development. We have standardized on Open Office because it has good bug tracking software, a responsive development organization, and a sufficient (and continually replenished) supply of bugs.

- • Working with a widely respected open source program allows our students to show off their work products during job interviews. There are no nondisclosure issues and the interviewer is unlikely to dismiss the work as unrealistic tinkering with a "toy" (academic) example. 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.

- • Standardizing on one application across courses allows us to teach one bug reporting and tracking process (in BBST-Bug Advocacy) and then rely on that knowledge when we teach later courses, in which students apply test techniques, find bugs, and report the bugs they find.

- • Situating student work in a respected, real-world project motivates students (Lave & Wenger, 1991) and 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). This is particularly important when dealing with working professionals because they are likely to ignore anything that is not obviously applicable to something that interests them (Knowles, Holton, & Swanson, 2005).

- • 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. The BBST-Bug Advocacy assignment (Appendix D) illustrates this. In the most desirable case, a few students submit work that is blasted as incompetent by an irascible programmer or project manager. (Open Office has some of these.) 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.

All assignments must be completed and submitted before the start of exam week.

▪ *Final Exam*: We draw the final exam from the study guide questions. The typical exam has 3 short and 3 long essay questions. An undergraduate who has prepared well can answer such an exam in less than 45 minutes. We give the exam Sunday morning, with a completion deadline of Wednesday night (midnight, local time). The exam is closed book, though we have no way to observe the students to see whether they are consulting other sources or not. Based on student performance and on their responses in anonymous course evaluations, we believe that we have had only one case of cheating across the courses so far. However, we recognize that different dynamics are in play in undergraduate and graduate classes--we currently proctor exams in the academic courses.

▪ *Final Exam Peer Review*. In the final segment of the course, students grade two other students' exams, then grade their own and, if they choose to, submit improved answers. Student-assigned grades are interesting but there is so much variation across students that these have little influence on the final course result. They do serve as a summary of the grading student's impression of the answer, which is useful feedback for the student who receives the grades.

▪ *Course completion*. The Association for Software Testing is not a university. We are more interested in advancing our members' abilities, knowledge and skills from whatever starting point they were at than in requiring them to demonstrate success against all of our learning objectives. We don't refer to successful completion of the course as "passing" the course. Instead, students either complete the course or they don't. Many students don't complete the course because they were called away to deal with this week's crisis at work. Some of those students stay with the course, even submitting a final exam, but they are so distracted with their jobs (or their family or whatever) that their exam is clearly too weak. In general, it is our intent to say that someone "completed" the course if they were cognitively present and engaged throughout the entire course. Under our (AST) policies:

- A student who writes an excellent final exam will complete the course

- A student who writes a weak final (but doesn't cheat) but demonstrates that s/he gained insight during the peer grading and self-reflection process will complete the course.

- A student whose exam and peer review are both weak is missing so much that s/he will not be allowed to "complete" the course.

- If exam performance is ambiguous, the student's contribution to the assignment is the next most dispositive body of work. After that, we look at preparatory exercises and quizzes, not to compute grades but to see whether the student made a consistent effort and demonstrated some basic level of knowledge and insight.

- Our bias is toward letting students "complete" the course. The typical AST course is co-taught by three or four instructors. All of them must agree to a result of "did not complete" for any student who submits a final exam.

Despite what we think of as a lenient completion policy, about 40% of our students do not complete the course, including about 7% who stay all the way through the exam.

▪ *Course Evaluation*. Our evaluation is based on the Student Assessment of Learning Gains (SALG) (Northern Arizona University Office of Academic Assessment, undated; Student Assessment of Learning Gains," 1997). We customize this heavily and now distribute the survey through SurveyMonkey. A typical respondent might take a full hour to complete the survey. SALG differs from typical assessment forms by emphasizing the learning objectives of the course, questioning

what aspect(s) of the course (if any) helped the student achieve (or make progress against) a given objective. We have found this feedback extremely valuable.

BUILDING A COURSEWARE COMMUNITY: THE CHALLENGES

Our goal is to improve the state of the practice of software testing by improving the education of software testers. That's the goal that has motivated us to spend thousands of hours developing the current generation of materials. And that's the goal that keeps us committed to the idea that our materials should be available to the world for free. To put the objective bluntly, we want our car's fuel injector to work (there have been deaths because of defective fuel injector software: "General Motors Corp. v. Johnston," 1992; Minasi, 1999) no matter where the software was tested.

Our initial thought was to put our course videos and other materials on a website and let people download what they needed. We thought they were largely self-explanatory, especially the videos. And therefore, we could develop the materials and (with a little marketing and sufficient bandwidth) distribution would take care of itself.

In retrospect, that was a naïve expectation. For complex material, people need to try things out, talk about them with someone who understands their questions or ideas, practice and get feedback.

As we see it now, to achieve our goal, we have to build a community of instructors who can competently and enthusiastically teach from our materials and help us keep them up to date and evolve them so that they are increasingly relevant to the diverse communities we want to serve.

The core problem is that this is going to take a lot of work and much of that breaks relatively new ground. Because we are trying to develop higher-level skills in our courses, our instructional model and goals look more like academic teaching than commercial. Our students do homework and assignments. They take exams. According to their course evaluations, they spend about 12 hours per week on an AST course. However, our audience--the audience we are discussing in *this* paper, and that AST is attempting to serve--is commercial. (Kaner and Fiedler are *also* working on adaptation for the academic market, but that's a different sub-project, for a different paper.)

The typical commercial short course is compressed. Either an expert comes to the student's company and stays/teaches for a few days (then goes away) or the student flies to a public course, probably taught in a hotel, often with classes that don't even use computers because the logistics of having one computer per student can be so difficult to arrange at hotels. (Some commercial courses are offered at training facilities that provide the equipment.) The course is filled with lecture or carefully planned activities. There is no time to try things, to try to apply things to the student's own work environment, or to work through the confusing parts of the lecture. After 8 hours in the classroom, the typical student has to catch up on their job (all the work they missed while they were in class) or fulfill family obligations at night. In contrast, the typical academic schedule allows slack time that students can use to explore the material critically, in more ways, in more detail.

All of these interaction with the students take time. Even though almost all student work is peer-reviewed in the AST courses and the lectures and many other instructional support materials are available, teaching this course takes so much time that the Association for Software Testing assigns three or four instructors to share the burden of each task. We are still learning how to improve our efficiency, but our impression is that at this time, the typical workload for a typical lead instructor or assistant lead instructor is about 12 hours per week. This is a very significant commitment.

Figure 2 draws more of the contrasts between academic and commercial short courses.

Academic Course	Commercial Short Course
Local instructor, who the student interacts with several times. Students get to know the instructor.	Visiting instructor (or a stranger in a conference / training room in a hotel). Few students get to know the instructor.
Spread over several months. Students have time to question and digest the material.	Rapid-fire ideas over a few days.
Deeper coverage of the materials	Broader, shallower coverage
Many courses emphasize activities expected to develop skills	Time constraints limit activities. Most courses rely heavily on lecture
Extensive homework	No time for homework
Students expect assessment (e.g. assignments and exams that are not trivially easy)	No exams, or a relatively easy multiple-choice exam
Coached, repeated practice is highly appreciated, especially if this material might appear on an exam	Coached, repeated practice seen as time-wasting. Coverage (more material) is more important than mastery. The expectation is that if a student sees that an idea or area is interesting, s/he will investigate it later, on her own time.
The goal is to develop capability (can the student DO this?)	The goal is to develop familiarity: Does this student know about this elect
Students have no work experience, need context	Work experience helps to bring home concepts
Harder to connect to the course to real practices in the field in a way that hits home for the students	Students have grounding in real practice and compare the course lessons with their experiences.
Students don't naturally come to a course as a group with a shared problem and therefore there is no natural application or task that all of them will want to solve.	Some (occasional) student groups share a genuine, current need. If all the students are in the group, the instructor can customize the course to help them with their issues.
Expect mastery of several concepts and skills	Objective: a few useful ideas that the student will consider applying on the job and exposure to a broadening set of definitions and ideas.

Figure 2: Contrasting Academic and Commercial Short Courses

In thinking through a strategy for attracting and retaining instructors and course maintainers / developers, we identified ten core challenges. We describe the challenges themselves in this section, and AST's approach to dealing with them in the next.

- *Challenge 1: Training Model*. How *should* we train working professionals? The course model of Figure 1 is certainly one approach, but some companies have already approached us to ask for other models. For example, some companies have asked us for a slower-paced course so their staff can spend 6 hours per week over more weeks, instead of 12 hours per week for 4 weeks. Other companies want live instruction, perhaps as a series of activity-intense short courses, or as a long boot camp. *Should AST create and maintain all these variations?*

- *Challenge 2: Adoption Model*. Why should people take the AST courses? Once we get past the early-adopter enthusiasts, what motivates the mainstream tester to work as hard as the AST courses

demand, for as many courses as AST provides? Other organizations motivate students by certifying them. The message they are selling is that someone with one of their certificates is probably more knowledgeable, more skilled, more professional, more expert than someone who is not certified. For example, the International Institute for Software Testing certifies students who have taken 10 testing courses (IIST teaches 1 day commercial short courses) (International Institute for Software Testing, 2008). As another example, the International Software Testing Qualifications Board (http://www.istqb.org/) offers certifications based on passing multiple-choice exams. Students typically prepare for these exams by taking a review course provided by someone tightly connected to ISTQB (for example, Rex Black, President of ISTQB, offers review courses for $2000-$2650. http://www.rbcs-us.com/Catalog/ProductList.asp?categoryid=270, accessed August 8, 2008). Will AST have to create a certification to foster adoption? Will anyone in the market for a certification be willing to work as hard as AST will require?

- ***Challenge 3: Development Model***. Kaner has done almost all the development of the current set of courseware. His knowledge has limits--we need other subject matter experts. His appeal has limits. A more diverse set of instructors, including lecturers on video, would provide a more diverse set of role models for new people entering the field. Thus, one set of development questions is who does the lecturing, the videotaping and editing, the development of quiz questions (which take us about 1.25 hours each, including time to write feedback for wrong answers), essay questions, assignments, preparatory exercises, selection of supplementary readings, creation of the Moodle pages, etc. How should we cluster and divide responsibilities?

- ***Challenge 4: Business Model***. Even though the course materials are free, the instructional support takes time. At the moment, we estimate 12 hours per instructor per week, for 6 weeks (including the week before classes start and the week after the course has officially ended). Should AST charge for this time? Should individual instructors? Under what circumstances?

- ***Challenge 5: Maintenance***. Who should update the slides, the videos, the quiz questions, and so on? What process should we follow to ensure that the right things get maintained.

- ***Challenge 6: Recruitment***. How should we recruit and qualify potential instructors for the BBST courses?

- ***Challenge 7: Instructor Training***. How should we train potential instructors? What training do they need? Who will provide it? How should we tell the world that person X has (or has not) been AST-Certified?

- ***Challenge 8: Instructor Retention***. How can we retain trained instructors? Should AST-certified instructors be required to teach for AST for free? Would this affect retention?

- ***Challenge 9: Funding***. How should we pay for the equipment? Should we pay anyone else to help teach, or help develop or maintain the courseware?

- ***Challenge 10: Intellectual Property***. Who should own the courseware? Should we (AST or Florida Tech) get a royalty every time someone distributes our product? Can anyone maintain the courseware?

We should note that even though the details of this list are our creation, our underlying ideas about the sustainability and potential profitability of open source projects come from detailed discussions among lawyers drafting legislation (primarily the Uniform Computer Information Transactions Act) over a period of six years, from direct discussions with participants in successful projects and from meeting (as counsel) with a few individual consultants who provide support services to the open source community. (Raymond, 2001) certainly influenced our thinking and we would welcome pointers to other detailed discussions of the business/organizing models that underlie some open source software communities, but at this point in this paper, we can only say that our analysis stands on broader shoulders than our reference

list reflects. We think we are distilling common knowledge from the open source software community that hasn't been written down in quite the right way to be useful as a reference.

BUILDING A COURSEWARE COMMUNITY: THE AST APPROACH

Consider a product like Linux. It is distributed for free--or at least under a free software license. In practice, many copies of Linux do go out for free, but many others are packaged for resale, such as Novell's distribution of SUSE Linux and Red Hat's Enterprise Linux. The customers for these distributions are willing to pay for a "free" product in order to obtain documentation and/or support.

Many individual consultants, small firms, and some larger firms (Red Hat is the most visible example to us) provide additional Linux-related services for a fee. People build their technical credibility and visibility by donating time, code, and support services to the free Linux project and then capitalize on this by selling development, support or training services to Linux-using companies. This is a symbiotic relationship: it provides an income stream for technical experts to support the ongoing development and maintenance of Linux itself and of the Linux customers.

Our vision ("our" includes the Association for Software Testing), is to apply this basic model to free courseware. We will develop courseware and distribute it under a Creative Commons license. We will:

- teach testing courses for free to our members (AST's dues are a modest $50 per year, rising to $85 in September, 2008);

- provide an instructor's course and many other instructor-support materials to members for free; and

- mentor instructors-in-training and provide a process to certify some of them as AST-certified instructors for specific AST courses.

However, we cannot teach enough courses enough times to satisfy likely demand, nor can we customize the courses for companies who want to run their testing staff through a set of AST courses. These companies can work around AST's limitations in three ways:

- Download the course materials, customize them as desired, and teach the course to their staff using their own trainers.

- Send some of their staff to AST to get instructor training and then have those trainers modify the courses as needed and teach them.

- Contract with AST-certified instructors, who teach the courses.

As we'll explain below, the second and third of these can provide resources back to AST.

Returning to free/open software, some projects (Eclipse, Firefox and Open Office are the examples with which we are most familiar) employ full-time staff for development and management activities, paying for them with funds from corporate or federal grants. The paid staff cannot do all of the work, but they can provide a stable core for coordinating and integrating contributions from volunteers. Grant-based funding is also part of AST's outlook.

In terms of the ten challenges, here are some details. These ideas are still incomplete. This project is a work in progress. But here is where we are now.

Challenge 1: Training Model

For now, all of AST's BBST courses will follow the 4-week online structure.

Companies have already contacted AST, or us individually, asking for customized versions or restructured versions of the course. As the pool of AST-certified instructors grows, we expect that several of them will offer corporate variations of the AST courses as part of their portfolio of consulting services.

Challenge 2: Adoption Model.

Why *should* people take the AST courses?

The primary reason is that these are good courses. Students rate them well. Our end-of-course survey asks students to compare the course to the difficulty and value of the commercial and academic courses they have taken. Students rate these courses harder and more valuable than their other commercial courses and comparable to academic courses.

AST members can take these courses for free. Opinion leaders who take the course encourage others, by word of mouth and by blogs (for example, Barber, 2008; Byregowda, 2008; Drake, 2008; Fry, 2007; Irvine, 2007; Osman, 2008; Perrold, 2007; Soundararajan, 2007).

After several requests, AST recently decided to give a Certificate to any student who completes 10 of the BBST courses. This will take much longer to obtain than other industry certifications, because this requires 40 weeks of course attendance, not 3 days and a quiz or 10 days. However, once some testers are certified, they will probably make themselves visible in ways that encourage employers to hire AST-certified testers and testers to take more courses.

The main impact of tester certification is this:

- Only AST-certified instructors can teach a course that counts toward the 10-course requirement, and

- A BBST course taught by an instructor certified to teach it will count for AST's 10-course credit, whether the student takes it through AST directly or somewhere else (e.g., at work).

Challenge 3: Development Model.

We need more course developers. Not much of what we have tried so far has succeeded. Here are plans for the future:

- AST recently formed an Education SIG, which has about 25 members, most of whom have expressed willingness to take on tasks. Now that we have detailed multiple-choice question writing standards (Kaner, 2007), we can try (again) to train some volunteers to draft good questions. So far, of the approximately 140 questions used in BBST Foundations and BBST Bug Advocacy, about 15 came from volunteers (the rest came from Kaner).

- We attempted to videotape other instructors, but we were going for chunks of work that were too large. We now expect to videotape short clips, presentations lasting a few minutes, that we can add as supplements to a main course. This will help train the instructor for video lectures and we expect that over time, we will get longer clips and a few courses.

Challenge 4: Business Model.

AST has adopted the open source model summarized above. When a company asks AST for a custom course or for a course that will be taught specifically for that company's staff, AST will send the company a list of AST-certified instructors who the company can retain for this purpose.

Additionally, much of this course development, especially the Instructor's Course, is being subsidized by a National Science Foundation grant to Cem Kaner and Rebecca Fiedler (Kaner & Fiedler, 2007a).

AST's capacity for teaching free courses is finite. Our goal is to teach, eventually, three courses in parallel every month. (This requires nine instructors, spending 12 hours a week each, every month.) We limit class sizes to 20 students, so in a given month, we can accommodate 60 students and over a year, 36 courses (720 students). If we have a portfolio of 15 courses, and 36 slots to teach them in per year, we will teach Foundations and Bug Advocacy several times per year and the others less often. People who want to take a course that is not offered at a convenient time by AST might take one instead that is

offered independently by an AST-certified instructor for that course. The instructor might offer the course for a fee, or in-house at the instructor's (and the student's) company, or as a free offering for the public or for some friends because the instructor chooses to do that.

AST's Executive Board recently decided that AST would *not* offer custom courses itself but would instead let the Certified instructors handle them.

AST certifies instructors on a course-by-course basis. For example, Scott Barber is certified to teach Foundations but not (yet) Bug Advocacy. One of the responsibilities of being certified is that the instructor has to teach the course (each course for which s/he is certified) for AST for free at least once per year. This is how we expect to maintain our instructor pool over time.

Challenge 5: Maintenance.

At this point, Kaner and Fiedler do almost all maintenance of the core materials of every course. This is unsustainable.

Suggestions for improvement are posted in the BBST Instructors Forum (Fiedler, 2008a) as are suggestions for standard posts by instructors to students, which we call Fieldstones (Fiedler, 2008d; Weinberg, 2005).

Challenge 6: Recruitment.

At this time, we recruit new instructors by invitation. We invite outstanding students from one of the courses to become an instructor for that course.

For new courses, which have not been taught before, we invite experts who have taken AST-BBST Foundations to help us teach the course the first times.

Challenge 7: Instructor Training.

- How should we train potential instructors? What training do they need? Who will provide it? How should we tell the world that person X has (or has not) been AST-Certified?

The next pages reprint the Association for Software Testing's plan and policies for training and certifying instructors (Association for Software Testing, 2008b).

Note that AST-certified instructors *must* be AST members and must maintain membership or lose their certification. AST's main reason for this is that membership in AST includes agreement to AST's Code of Ethics (Association for Software Testing, 2006, currently identical to ACM's Code of Ethics). One of AST's priorities for 2009 is establishing an enforcement process for the Code of Ethics. AST-certified instructors who were accused of conflict of interest violations or other unethical conduct would be subject to review and discipline under this process.

As a summary of the process, a typical instructor will teach the course three (or more) times under supervision of an experienced lead instructor, will take the BBST Instructors Course and submit her or his work products to the already-certified instructors for this course, and will be accepted as a Certified Instructor for the course if and only if these instructors agree that the candidate is good enough.

Association for Software Testing Certification of Instructors

Figure 3: Becoming an AST-Certified Instructor

Becoming an AST-certified Instructor
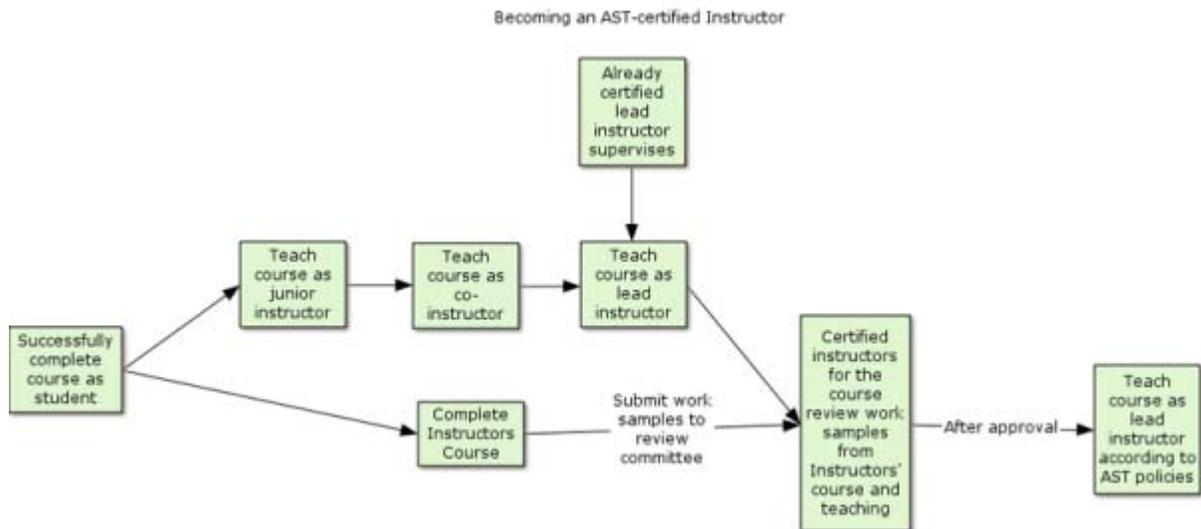
- AST will certify instructors to teach the AST courses. Certification is on a course by course basis. Thus an individual may be certified to teach some AST courses but not others.
- The AST Course Administrator has discretion to certify an instructor for an AST course (we will refer to this as "grandfathering"). The intent of this certification is to create one or two certified instructors for a new course who can then train additional instructors. It should be unusual, and cause for an AST Executive Board query if the AST Course Administrator grandfathers someone for a mature course (one that has a few instructors and has been taught a few times). However, even for a mature course, the AST Course Administrator has discretion to certify herself or himself. In all cases, including self-certification by the Administrator, the Course Administrator must be fully confident that a candidate for grandfathering has the knowledge, attitude and skill needed to teach the course well.
- Anyone who wants to be AST-certified to teach Course X must do all of the following:
  o take Course X
  o teach Course X three times under supervision of an AST certified instructor, with increasing responsibility each time, and as lead instructor the third time
  o complete the instructors' course
  o submit the performance components of the instructors' course (grade sample exams, provide feedback on simulated discussions, etc.) to the current instructors of Course X for their review
  o be approved by all of the current instructors, after their review of the candidate's teaching of Course X and performance on the instructor course. We will revise this percentage down from 100% later, when there are many more instructors.
- Once an instructor is certified to teach one course, s/he can obtain certification for a second course (Course Y) more easily. S/he must:
  o take Course Y as a student
  o teach Course Y twice under supervision, once as lead
  o be approved by all of the other Course Y instructors
- Anyone can take the instructors' course, which will be posted publicly and self-paced.
  o The primary assessments will be multiple choice. This will be for the feedback to the student.

17

- o There will be optional performance tests that practice the instructor-trainees on grading and other skills.
- o No one will grade the performance tests, but students will be allowed to make a copy of their work for review by someone else (e.g. by the Course X instructor pool). Students will get little or no feedback on any of their performance from anyone assigned to monitor the Instructors' course. The instructor course will not have an active facilitator in the way that the BBST courses do.
- o Students are responsible for keeping their own copies of their results. We will erase student results from the website on an ongoing basis (say, once a month).

Responsibilities of Instructors

- An instructor certified to teach Course X must teach Course X to AST students (as an unpaid instructor for AST) at least once per year. However, no matter how many courses an instructor is certified to teach, s/he is not responsible for teaching more than 3 courses for AST per year.
- Instructors must recertify every three years. Recertification is by a super-majority vote (70%) of the other instructors certified to teach the course. The vote is to be based on performance of the instructor when teaching the course. The recertification candidate will give the other instructors access to archives of courses s/he has taught for AST, and possibly other courses.
- AST-certified instructors must keep their membership in AST in good standing. A person who leaves AST is no longer AST-certified to teach AST courses.
- Instructors must respect the limited confidentiality of student data and all of the other aspects of the course agreements that all students and all instructors are required to agree to in each course.
- The instructors of a course must do all of the tasks required for the course, including preparing to teach the course, providing feedback in forums, grading, reporting results to AST, and setting up the course evaluation.
- As a condition of certification, AST-certified instructors must agree in writing that they understand AST certification to be a privilege and not a right, and that they agree to the disciplinary process outline below:
  - o The AST Executive Board may revoke an instructor's privilege to teach courses as an AST-certified instructor at any time and the instructor agrees to accept such a decision as binding and final.
  - o Decertification shall be a private matter unless:
    - ▪ The decertified instructor continues to state publicly (orally, on her or his website, in her or his printed materials or in any other way) that s/he is AST certified or that a course s/he teaches is accredited by AST. In this case, the AST Executive Board shall publish a notice that this instructor is not certified and that courses in which s/he is the lead instructor are not AST-accredited and cannot be accepted for credit towards the AST certificate.
    - ▪ The decertified instructor states publicly, in a way that is likely to be seen or heard by many people (for example, on a blog posting or in a conference presentation) or in any legal proceeding (in a court of law or in arbitration) that s/he was treated unfairly by AST. This shall be interpreted as a waiver by the instructor of all of her or his privacy rights in her or his performance or conduct as a student or instructor in AST-accredited courses. In this case, the AST Executive Board may choose to publish a rebuttal, but such a rebuttal must be considerate of the privacy of other parties. It would not be unusual for AST to consult counsel before publishing a rebuttal, nor would it be unusual for the AST Executive Board to choose not to respond or to respond in a very generalized way. If such a statement is made only in a private arbitration or in private correspondence with AST or a mediator interacting with AST, then AST shall respond in a way that appropriately protects its rights and reputation but that does not make the matter more public than its reasonable perception of the extent to which the instructor has made it public.
- The AST Executive Board delegates to the AST Course Administrator the primary responsibility for dealing with problems in the teaching of courses. In particular, the AST Course Administrator will suspend the certification of an instructor appears to be violating the AST Code of Ethics, course-related rules or agreements, or failing to live up to responsibilities, The AST Course Administrator will then convene, at a

convenient time and without unreasonable delay, a recertification board meeting to determine the appropriate action.

- o The recertification board is composed of the other instructors certified to teach this course, plus the AST Course Administrator. The board may also invite or retain advisors to attend, provide counsel, but not vote at their meeting(s).
- o A member of the recertification board who has a direct financial interest in the result shall identify her or his interest and shall be recused from the meeting (and all other formal or informal deliberations)UNLESS NEITHER the instructor NOR the AST Course Administrator object to that member's continued participation. A member of the recertification board who has an emotional response to the instructor that involves matters outside of her or his conduct as an instructor, and who feels that there is a nontrivial risk that this response will bias her or his judgment in this matter MUST recuse herself or himself from participation in the board meeting and all other formal or informal deliberations on this matter.
- o The recertification board may meet in person, by conference call, or electronically, or in some combination of all three. The board "meeting" may be spread over several days.
- o The recertification board meeting must include a period in which the instructor is allowed to be present, and this portion of the meeting must be scheduled at a time that the instructor can attend. The instructor may speak, provide or examine documents, and ask or answer questions during this portion of the meeting.
- o The instructor does NOT have the right to confront or question a complaining student if that student chooses to remain anonymous. However, the AST Course Administrator must summarize complaints that will be considered by the recertification board, along with her or his assessment of their credibility and significance.
- o The recertification board may also include a period for private deliberation (deliberations open only to the recertification board and its advisors)
- o The primary consideration of the recertification board should be protection of the rights, safety, and quality of instruction of the students. Protection of the instructor's right to continue teaching as an ASTcertified instructor is an important consideration but secondary to protection of the students. Protection of the AST is also a very important consideration for the recertification board. AST will be best protected if the board follows reasonable procedures, operates fairly and with courtesy and consideration to all involved, and acts consistently from case to case. Beyond that, protection of AST shall be considered, but as a tertiary priority, behind protection of the students and fairness to the instructor.
- o An instructor my appeal the decision of the recertification board to the Executive Board of the AST. In doing so, s/he creates a limited waiver of her rights to privacy in her or his performance or conduct as a student or instructor in AST courses. Under this waiver, the AST Executive Board is entitled to gather all potentially relevant information, but any broader dissemination of private information is not authorized by the waiver.

Challenge 8: Instructor Retention.

How can we retain trained instructors? We haven't faced this problem yet. If the business model works, this probably isn't a problem.

Challenge 9: Funding.

How should we pay for the equipment? Should we pay anyone else to help teach, or help develop or maintain the courseware?

At this time, we are paying for everything either out of NSF grant funds, grants by private donors (Satisfice, Inc. has committed $10,000, for example), or (to a large degree) out of our personal funds. AST is providing server space and system administration support.

Challenge 10: Intellectual Property.

All of the AST course materials (except for assigned readings that we are using but that were not specifically created for the AST courses) are released under a Creative Commons license. At moment, Cem Kaner or Rebecca L. Fiedler are copyright holders for almost all of the material (everything except what is contributed in the instructor forum and wiki).

At the moment, most of the materials are provided under an Attribution/Share-Alike license that requires people who modify the course to share their modifications. We are revising this to an Attribution license (without share-alike) so that people can make modifications, teach their own courses to the public or at their company, without having to share their changes.

There is extensive debate within the free/open software community about whether share-alike (copyleft) is good or bad. We are not expressing an opinion here except to say that for our particular purposes, with our particular objectives of improving the state of the practice generalize, the more courses that are improved by a judicious sampling from our materials with enhancements suited to those courses, the better. Anything that stands in the way of this, as copyleft would for some instructors, is contrary to our objectives.

ACKNOWLEDGEMENTS

Correspondence concerning this paper should be addressed to Cem Kaner, Computer Science Department, Florida Institute of Technology, 150 West University Blvd, Melbourne, Florida 32901, E-mail: kaner@kaner.com

APPENDIX A. THE FIRST PREPARATORY EXERCISE IN BBST-FOUNDATIONS

Please answer these questions BEFORE viewing the lecture on the oracle problem. After you submit your answer, please post comments on at least two other answers.

These questions serve two key purposes:

- they set a better foundation for learning, by raising some issues that you think about first, before you see how they are addressed in the lecture.
- they help us understand what the class knows, so we can lead the course more effectively.

Suppose you were writing a text editing program that would display fonts properly on the screen. How would you test whether the font display is correct?

1. How could you determine whether the editor is displaying text in the correct typeface?
2. How could you determine whether the editor is displaying text in the right size?
3. How would you test whether the editor displays text correctly (in any font that the user specifies)? Describe your thinking about this problem and the strategies you are considering.

Here are some notes on terminology:

A *typeface* is a style or design of a complete set of printable characters. For example, the following are all in the typeface "Times New Roman":

Times New Roman, 8 points

Times New Roman, 12 points

Times New Roman, size 18 points

Times New Roman, size 24 points

The following are in the Helvetica typeface:

Helvetica, size 8 points

Helvetica, size 12 points

Helvetica, size 18 points

Helvetica, size 24 points

When you add a size to a typeface, you have a font. The line above this is printed in a Helvetica 24 point font.

APPENDIX B. BUG REPORTING EXERCISE FROM BBST-BUG ADVOCACY

This exercise reviews a lecture on troubleshooting and previews a lecture on bug reporting style and clarity.

The assignment presents two distinct failures, that might or might not come from the same underlying cause, and asks students to write the summary line (one-line title) for the bug report. We have already encouraged the students to write two bug reports when they see too failures, but this (and other issues of clear reporting) will come up in greater detail soon.

The students who do attempt to file only one bug report write vague summary lines like "Failure when cutting" or "Problem with zoom and cutting."

The exercise feedback (and lecture) emphasize the "Fails -- When -- protocol" (tell me how it failed, then tell me when / how (critical conditions) it failed and the need for specificity.

Examples of better summaries are:

- First failure: "Cut doesn't cut. (Freehand select, zoom)"
- Second failure: "Cut the wrong area. (Freehand select, zoom, resized window)"

Bug Reporting Exercise

The following group of slides come from Windows Paint 95. You won't be able to replicate this bug in recent versions of Windows Paint because it's been fixed.

Please treat all of the steps shown in these screen shots as if they were fully reproducible.

In case you aren't familiar with paint programs, the key idea is that you lay down dots. For example, when you draw a circle, the result is a set of dots, not an object. If you were using a draw program, you could draw the circle and then later select the circle, move it, cut it, etc. In a paint program, you cannot select the circle once you've drawn it. You can select an area that includes the dots that make up the circle, but that area is simply a bitmap and none of the dots in it have any relationship to any of the others.

SCREEN 1

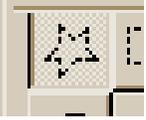Here's the opening screen. The background is white.
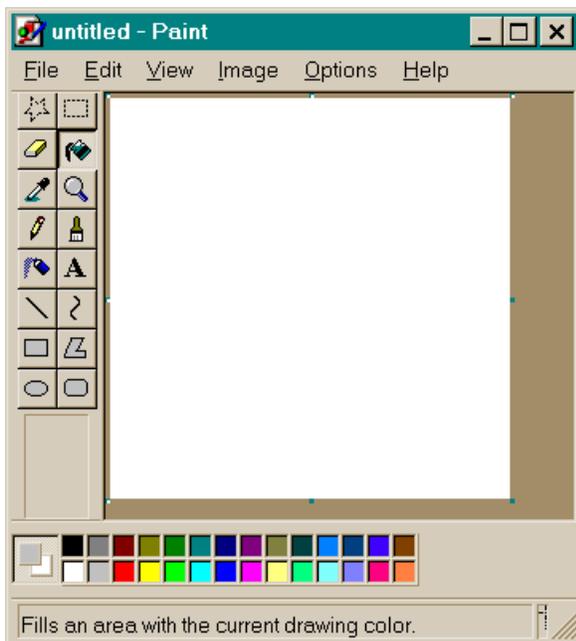
The first thing that we'll do is select the Paint Can

We'll use this to lay down a layer of grey paint on top of the background. Then, when we cut or move an area, we'll see the white background behind what was moved.

SCREEN 2
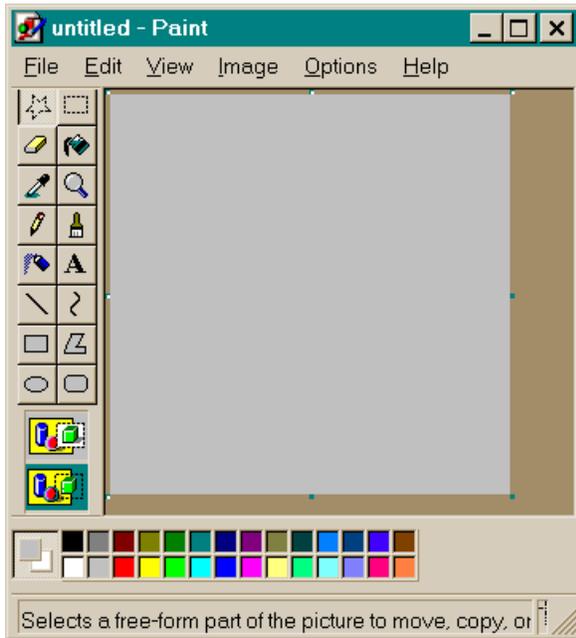
The star [icon] in the upper left corner is a freehand selection tool. After you click on it, you can trace around any part of the picture. The tracing selects that part of the picture. Then you can cut it, copy it, move it, etc.

SCREEN 3
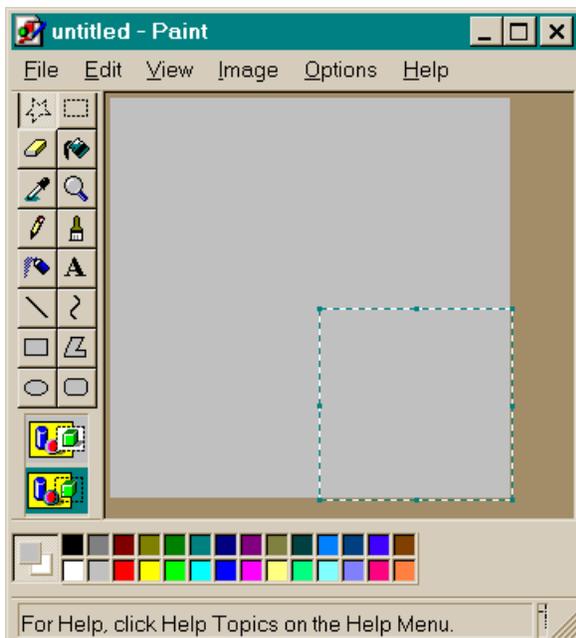
This shows an area selected with the freehand selection tool. The bottom right corner is selected. (The dashed line surrounds the selected area.)

NOTE: The actual area selected might not be perfectly rectangular. The freehand tool shows a rectangle that is just big enough to enclose the selected area. For our purposes, this is not a bug. This is a design decision by Microsoft.
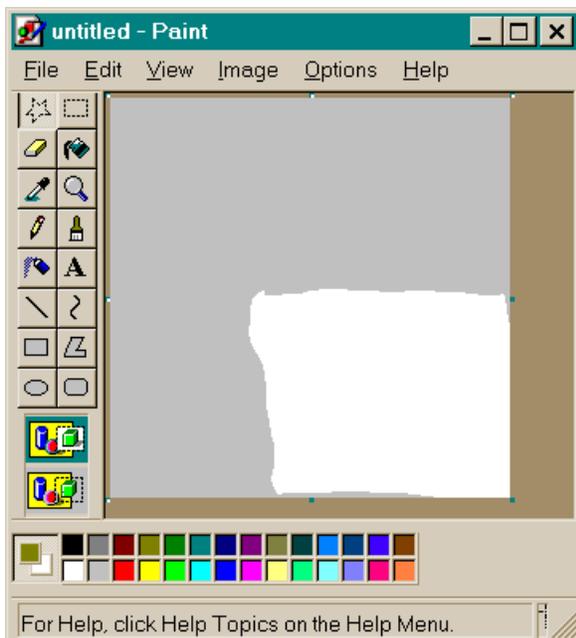
SCREEN 4
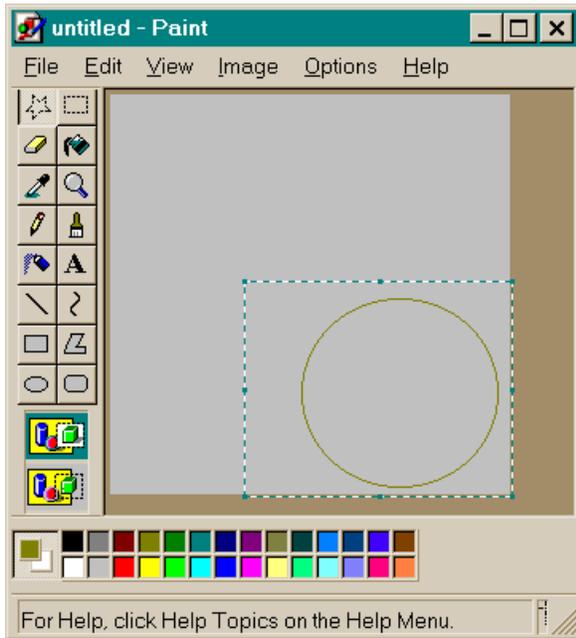
Now, we draw a circle (so you can see what's selected), then use the freehand select tool to select the area around it.

When you use the freehand selection tool, you select an area by moving the mouse. The real area selected is not a perfect rectangle. The rectangle just shows us where the selected area is.
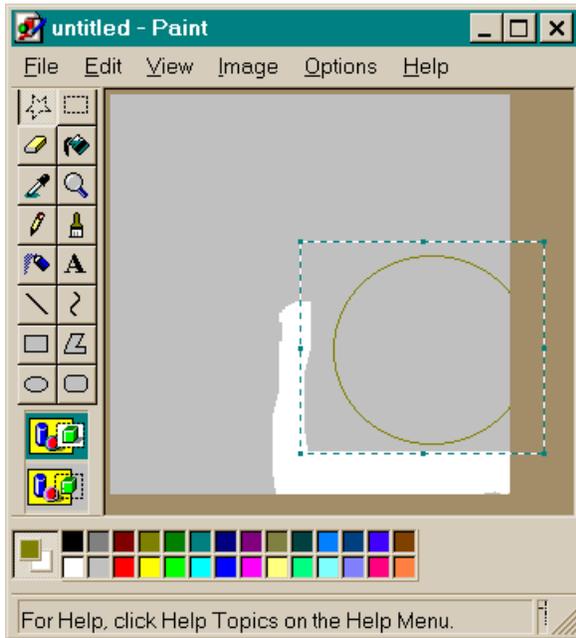
SCREEN 5

Now we cut the selection. (To do this, press Ctrl-X.)

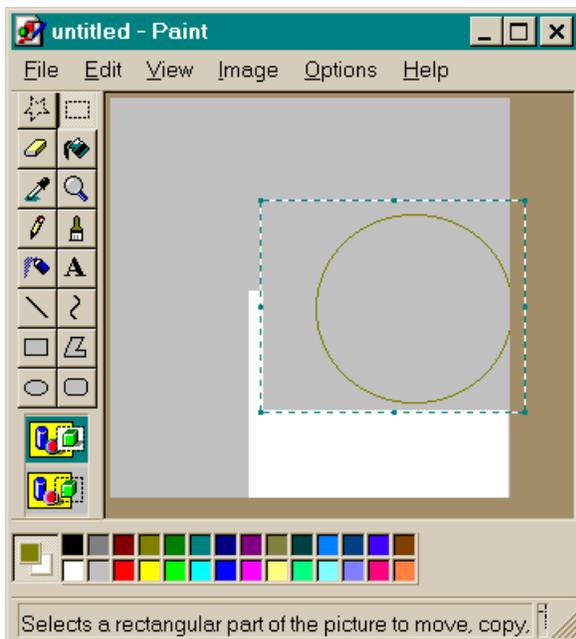The jagged border shows exactly the area that was selected.

SCREEN 6

Next, select the area around the circle and drag it up and to the right.

This works.

Now, exit the program, start the program again, paint the screen grey, and draw the circle in the lower right. **We'll do these steps with every test from here on.**

SCREEN 7

This time, we'll try the Rectangular Selection tool.

With this one, if you move the mouse to select an area, the area that is actually selected is the smallest rectangle that encloses the path that your mouse drew.

So, draw a circle, click the Rectangular Selection tool, select the area around the circle and move it up. It works.

Well, this was just too boring, because everything is working. When you don't find a bug while testing a feature, one tactic is to keep testing the feature but combine it with some other test.

In this case, we'll try Zooming the image. When you zoom 200%, the picture itself doesn't change size, but the display doubles in size. Every dot is displayed as twice as tall and twice as wide.
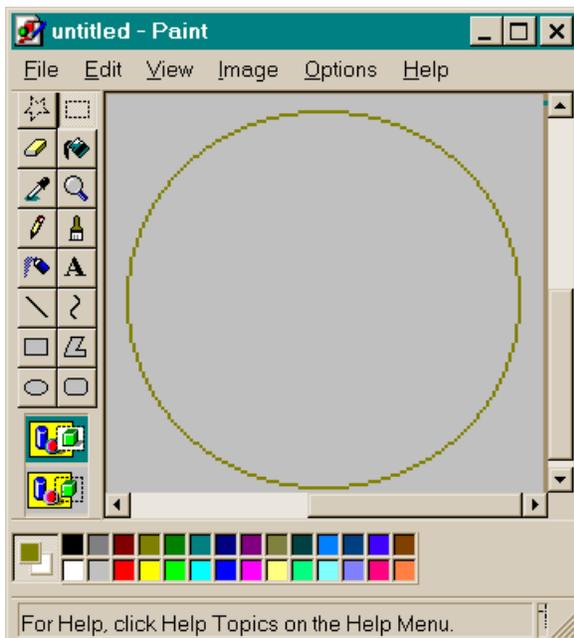
SCREEN 8
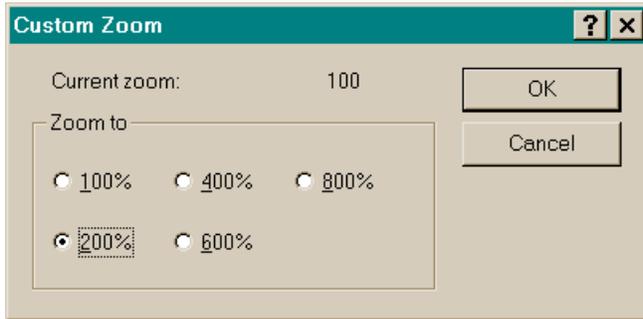
Bring up the Custom Zoom dialog, and select 200% zoom, click OK.

Our standard test sequence now starts like this: Exit the program, start the program again, paint the screen grey, draw the circle in the lower right, and zoom 200%. **We'll do these steps with every test from here on.**
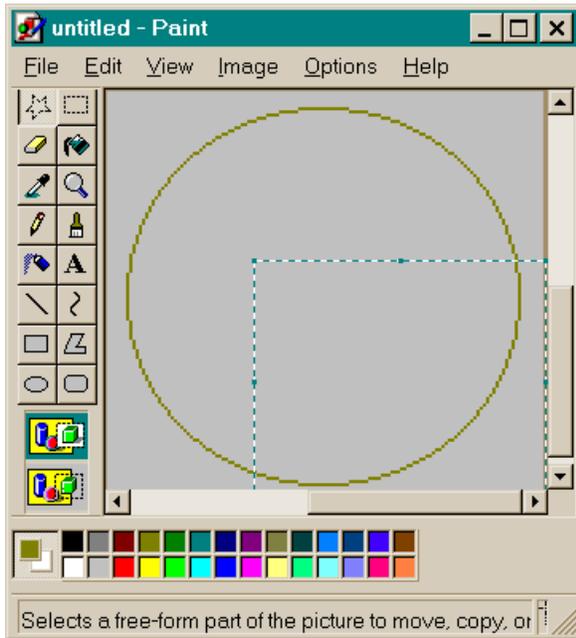
SCREEN 9

It worked. The paint area is displayed twice as tall and twice as wide.

We're now focused on the lower quadrant of the paint area, the circle that we drew in the bottom right corner.

To see the rest, we could move the scroll bars up or left.

SCREEN 10

Select part of the circle using the freehand selection tool. We'll try the move and cut features.

<u>Cutting fails.</u>

When we try to cut the selection (with CTRL-X), the dashed line disappears, but nothing goes away.

SCREEN 11
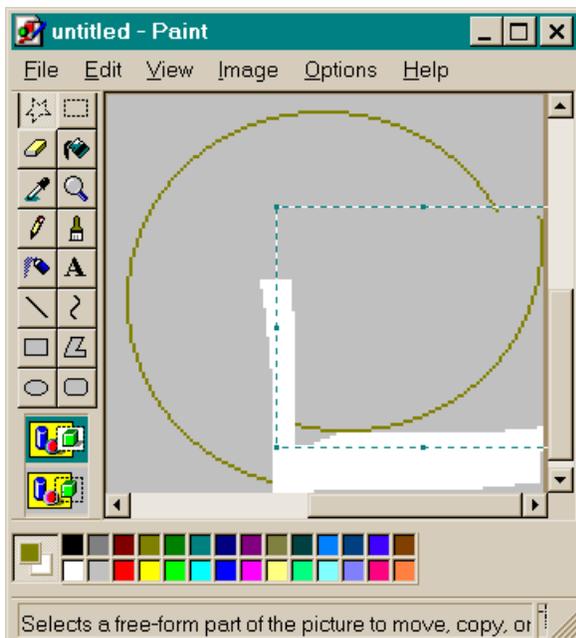
Let's try another test. This time, we select the area, but then instead of trying to cut it, we drag the area up and to the right.

That works.

SCREENS 12 and 13
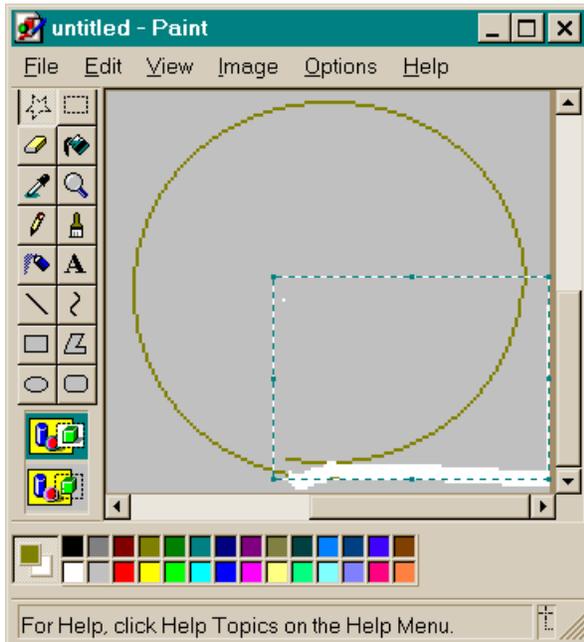
In this test, we select the area and move it a bit.
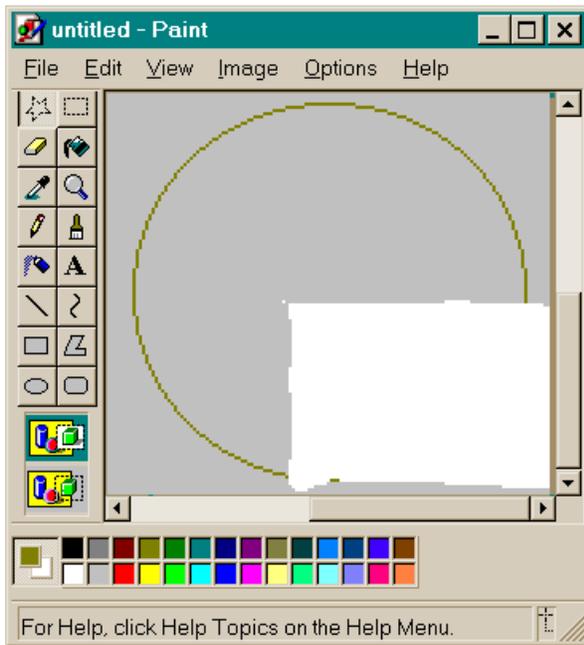
THEN press Ctrl-X to cut.

This time, cutting works.

SCREEN 14
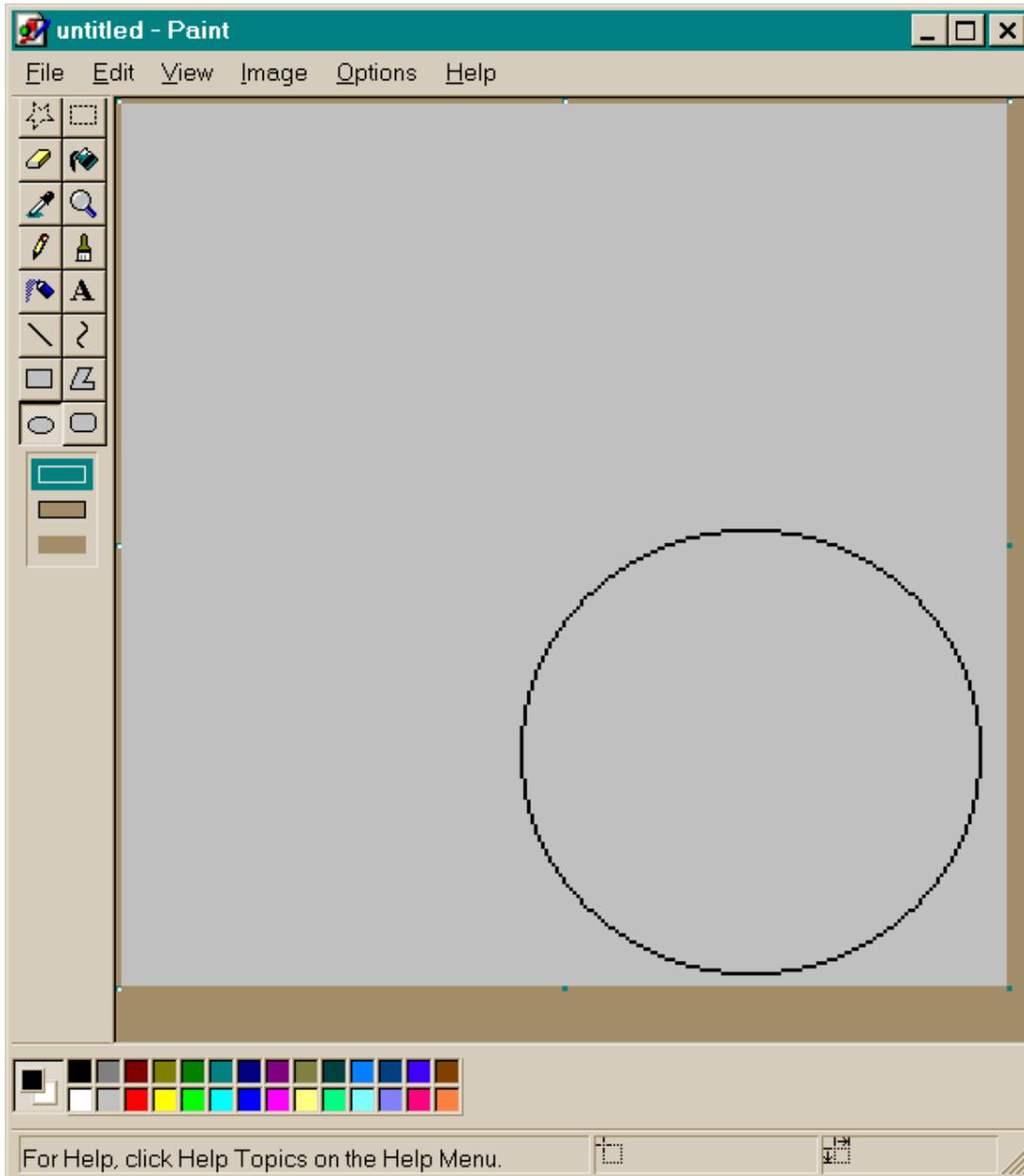
The last result was odd, but it's hard to tell what happened. Cuttingworked, then it failed, then it worked. In the last few tests, after zooming, we were only looking at the bottom corner of the paint area. Let's stretch the window (maybe we have to move the test to another machine with a bigger monitor) so that we can see the entire paint area.

SCREEN 15

untitled - Paint

File   Edit   View   Image   Options   Help

For Help, click Help Topics on the Help Menu.
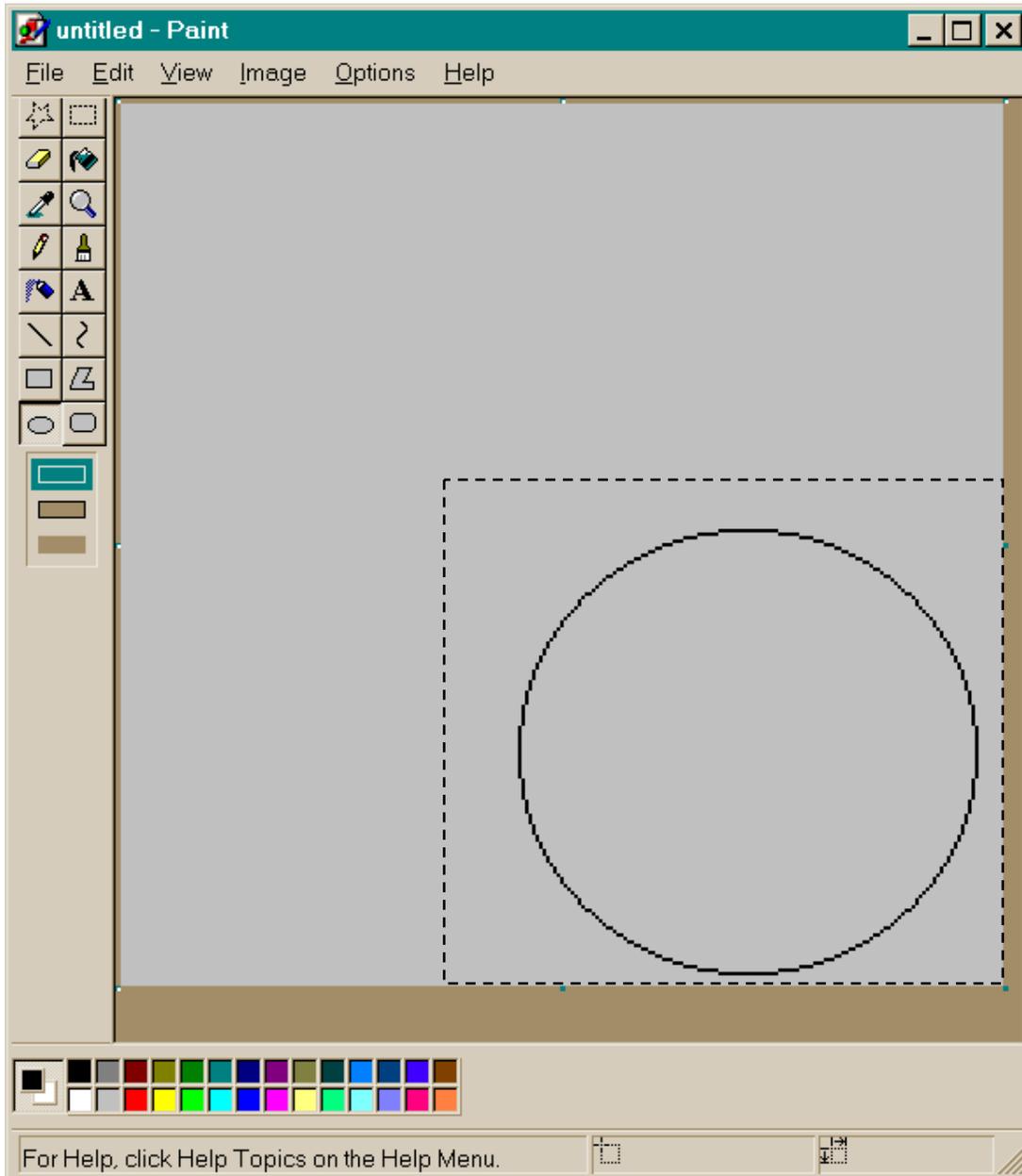
Selecting the circle seems to work

SCREEN 16
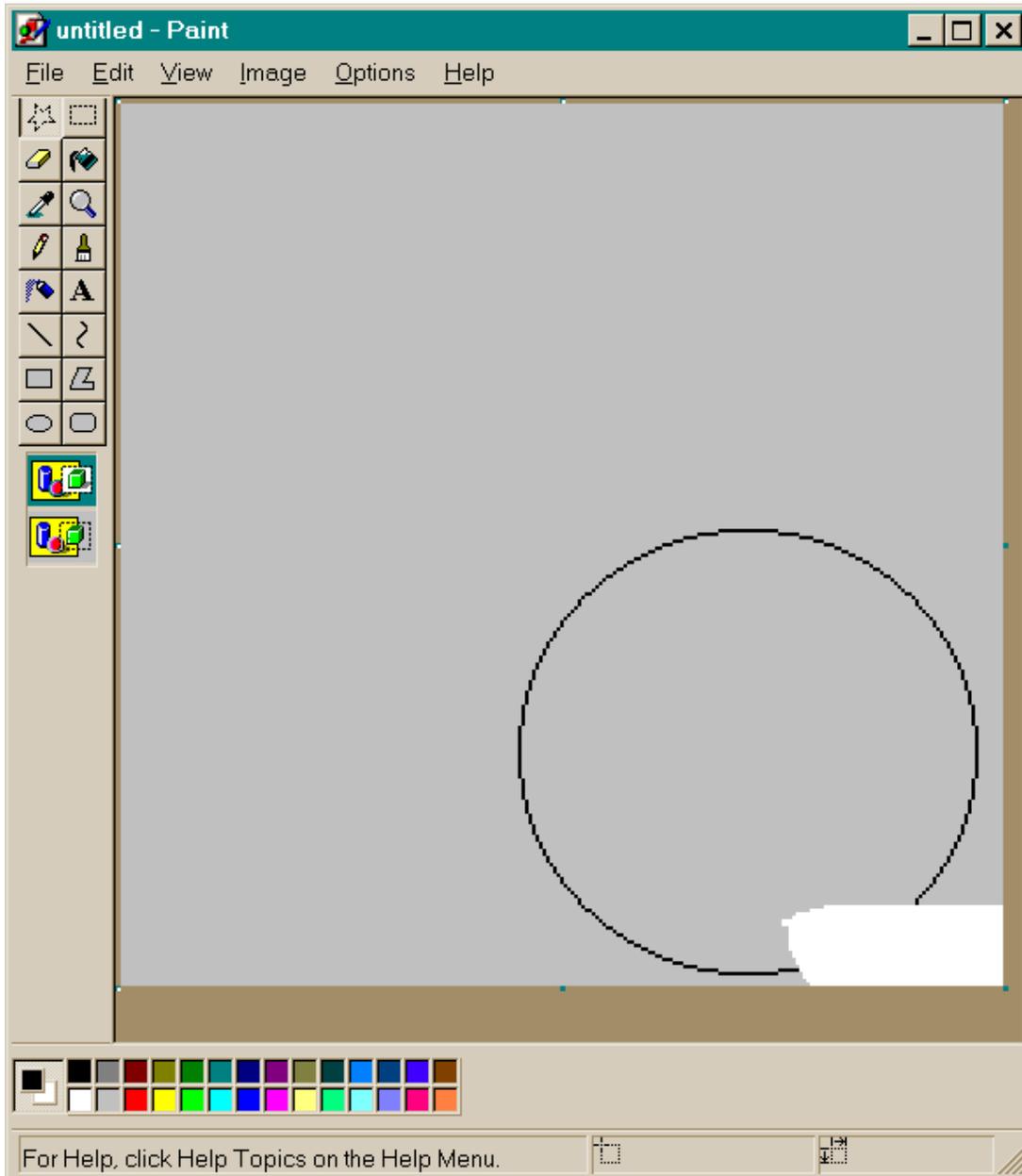
But when you press Ctrl-X to cut the circle, the program cuts the wrong area.

YOUR TASK

1. Suppose this is all of the investigation that you have time to do before reporting this behavior. Write and submit the Problem Summary for the bug report. (The problem summary is a 50-70 character (8-12 word) description of the problem. Many people will see ONLY this part of your bug report.)

1a. Review and critique summaries from two other students

2. If you had time to do more follow-up tests, what would you do, and why?

2a. Review and critique the follow-up suggestions from two other students

You might find the following table useful for organizing your thoughts.

Observed Failure	Critical conditions
Other possibly-relevant conditions	**Notes**

APPENDIX C. THE BBST-FOUNDATIONS ASSIGNMENT

Assignment: The Mission of Testing

Your instructor will assign you to a group. Your group will receive a private discussion board and a private wiki in which to complete your task. Your group will collaboratively develop a report using a wiki. (Are you unfamiliar with wikis? Take a look at this tutorial) Your instructor is a member of all groups to monitor efforts and contributions of all parties. Unless necessary, the instructor will not participate in the functioning of the group.

Your group's report is due by Midnight Wednesday of Week Three.

Project Description

You are testing a program that includes spreadsheet features. Your work involves testing those features. Your report should consider the following questions, and any others that you think should be added.

1. What is your mission?
2. How should you organize your testing to help you achieve the mission? For example,
    1. How aggressively should you hunt for bugs? Why?
    2. Which bugs are less important than others? Why?
    3. Suppose the program has a numeric input field. The spec says it must accept single digits, but not how it should
       respond to letters? Should you test with letters? Why or why not? What if you're time pressed?
3. How extensively will you document your work? Why?

The Groups

Group 1: Early Development

You have joined the development team very early in the project. The project manager has asked you to help her understand the product's risks so that she can manage the programming and later testing more effectively in terms of managing the reliability of the product.

Group 2: Late Development

You have joined the development team close to its release date. The project manager expects you to test the product in ways that tell her whether it is ready for release.

Group 3: Custom Software

You have joined the development team in a company that is doing custom software development, writing this program to a specification that was negotiated and incorporated into the development contract. Your objective is to test the program in a way that lets you determine whether it will be acceptable to the customer (as indicated by conformity with the specification).

Group 4: Medical Software

You have joined the development team for a spreadsheet component of a product that will be used in medical services, to track treatment history, prescriptions, etc. Doctors and nurses will rely on the information stored here when medicating or otherwise treating their patients. Your objective is to test in whatever way will most help the company get approval of the US Food & Drug Administration.

Group 5: Computer Game

You have joined the development team for a role-playing computer game. The spreadsheet features will keep track of character attributes, for example the character's experience points, health, age, and equipment or spells it is carrying.

Project Logistics

Each group will "meet" in their private discussion forum to develop a plan for working on this project. You may organize yourselves in any way that you think will enable your group to work effectively.

We expect you will do all of the work in the discussion board and the wiki. We expect the final project to be presented in the wiki. To facilitate communication about entries, all contributions to the wiki will be signed by the contributor. Note that wiki technology allows the instructor to monitor revisions and contributors.

All members of the class are expected to participate in their respective groups and to communicate with them regularly.

Appendix D
Association for Software Testing
Black Box Software Testing Series: Bug Advocacy Course

Assignment – Replicate and Edit Bugs

Cem Kaner

Version 9.1, July 2008

Overview

The purpose of this assignment is to give you experience editing bugs written by other people. This task will give you practice thinking about what a professional report should be, before you start entering your own reports into this public system.

The guidelines that I provide for evaluating bug reports are based on criteria developed by several test managers to help them evaluate the quality of their staff's work.

Phase 1

You'll start by submitting comments on one UNCONFIRMED report on the IMPRESS (or Presentation) product. This includes:

- comments designed to improve the report—make these as your changes to the report stored in the OpenOffice tracking system

- comments designed to evaluate the report—make these as comments on our forum

Phase 2

- After you submit your comments, please peer review comments from two other participants in the course.

Phase 3

In the third phase, you will pair up with another participant. Pick one UNCONFIRMED report on IMPRESS each. These must be different reports and they should be different from the reports that class has already worked on.

In this phase, before you submit any comments to the OpenOffice database or to our forum:

- send your comments to your partner and get his or her feedback first. Then improve your work as appropriate and after that,

- file comments designed to improve the report with the OpenOffice tracking system

- file comments designed to evaluate the report on our forum

- file comments on how the peer review helped (if it did) and whether you think it was worth it (and why)

Phase 4

In the fourth phase, you will grade two of the reports submitted in Phase 3, using the grading guidelines that we provide.

Background to the Assignment

We want you to develop experience in evaluating and editing bug reports for a few reasons:

1. We have been repeatedly shocked by the low communication quality of bug reports at good companies. That makes it important for us to work on your technical and persuasive writing skills as they apply to bug reporting.

    a. The simplest form of training--having students write bug reports and then critiquing the reports, doesn't always work well. With many writers, it takes report after report after report to get much improvement, as it does in many other courses that try to teach writing skills.

    b. A different approach to writing instruction, which is what we're trying here, is to teach you to be critical, analytical readers of *other people's* bug reports. As you see the common mistakes in these reports, identify them and explain what makes them problematic, you will develop better skills for evaluating your own reports.

2. It is common for companies to require testers to review bug reports before sending the reports to the programmers"

    a. This is especially common on open source projects, because the people who write the original bug report are often inexperienced observers and writers. Sending first-draft bug reports to volunteer programmers, often wasting their time, will drive away many volunteers.

    b. Even some of the more traditional organizations have testers review bugs (before they go onto the queue for programmers), especially design requests, bugs from new testers, and bugs from non-testers. Some independent test labs claim to review every bug before passing it to their clients.

    c. As you'll see, this process has benefits and costs. It takes a lot of time. If you are going into testing, or development management, having personal experience with the costs and the benefits will help you assess your bug reporting process.

3. As a manager, you will often be asked to evaluate your staff. This is very difficult. Feedback to staff on the quality of their work is often weak.

    a. Detail-oriented reading, that starts from a carefully considered list of questions about the quality of a given task, can give you a solid, fact-based foundation for staff training and decisions about raises, layoffs, and promotions.

    b. The evaluation framework that we use in this assignment is derived from materials actually used by test managers to review their staff's work.

The challenge in following this approach in this course is that many students are also weak critical evaluators of other people's writing. This skill is not widely taught to science and engineering students, so along with asking you to apply this style of evaluation once, we are using this assignment to give you practice with the approach.

Our experience with variants of this assignment in the past suggests that:

- **Phase 1** asks you to find an unconfirmed bug in the OOo database, replicate it (if you can), add comments that improve the underlying report, and submit (to the class or the instructor) an evaluation of the quality of the report. We provide a 35-question framework, with questions divided into four categories, to help with the evaluation.

- **Phase 2** gives you the opportunity to see how other students handled the same task. Your evaluation of their work (using the same 35-question framework) will often lead to insights into both, what makes a bug report strong or weak, and what makes an evaluation strong or weak.

- Our **Phase 3** goal is for you to do a better Phase 1:

  - better contribution to the OOo bug report (submitted to the OOo database)

  - better evaluation of the OOo bug report (submitted to us)

  The Phase 3 tools to support that are:

  - peer review (private, between partners)

  - repeated emphasis on the 35 questions in the framework. If you are curious about the instructional history of student-guiding frameworks (like this one), read up on the [concept of scaffolding](#).

  Coming out of Phase 3, we hope you see a big improvement in your work and your partner's as a result of the peer review. This is the best way we know to demonstrate the value of having testers review each other's bugs.

- Our hope for **Phase 4** is that when you evaluate the output from Phase 3, you will see work products that are a lot better than you saw in Phase 1.

  - If the work products really are good, this evaluation will be very fast.

  - If the work products are not so good, they are probably still better than Phase 1, but this gives one last training for improving future bug reporting. For several students, this last review is when they finally "get it."

Good testing requires development of several skills. The idea of this multi-phase assignment structure is to keep at the task, approaching it from a few different ways, until you get good at it. Our experience suggests that several students will need to work through all four phases and the feedback they get from them to finally cross over into doing obviously good work without struggling to achieve it.

Details of the Assignment

Phase 1

You'll start by submitting comments on one UNCONFIRMED report on the IMPRESS (or Presentation) product. This includes:

- **comments designed to improve the report—make these as your changes to the report stored in the OpenOffice tracking system**

- **comments designed to evaluate the report—make these as comments on our forum**

Find several bug reports in the tracking system that have not yet been independently verified. These are listed in the IssueTracker database as UNCONFIRMED. *If you submit comments reports marked NEW or something other than UNCONFIRMED will be rejected as nonresponsive to the assignment.*

From here, you have two tasks:

1. *EDIT the report TO IMPROVE IT*: revise it in ways that help the OpenOffice developers prioritize the bug accurately and fix it efficiently. Be sure that your added comments add value, because every time you edit a bug report, IssueTracker sends an email to many members of the development team. Making a bunch of low-value edits to the bug reports is like making a denial-of-service attack on the time/attention of the developers. This can make them grumpy.

   Some common ways to improve the report include:

   - Add a comment indicating that you successfully replicated the bug on XXX configuration in YYY build.

   - Add a comment indicating that you were unable to replicate the bug, if that failure would be informative. For example, if troubleshooting so far indicates this is a Macintosh-only bug, successful replication on a Linux system would be interesting, but failure to replicate on a Linux system would be uninformative. If you are reporting failure, what variations of the reported replication steps did you try in your effort to replicate the bug.

   - Add a comment describing a simpler set of replication steps (if you have a simpler set). Make sure these are clear and accurate.

   - Add a comment describing why this bug would be important to customers. (This is only needed if the developers are likely to defer it because it seems minor or unimportant. It is only useful if you know what you are talking about).

   - Your comments should NEVER appear critical or disrespectful of the original report or of the person who wrote it. You are adding information, not criticizing what was there.

   If you edit the report in the database, **never change what the reporter has actually written**. You are not changing his work, you are adding comments to it at the end of the report

   Your comments should have your name and the comment date, usually at the start of the comment, for example: "(Cem Kaner, 12/14/09) Here is an alternative set of replication steps:")

2. *EVALUATE the quality of the report*—as a report. This does not go to the OOo developers, it stays with us. This task should help you think about how to write more effective bug reports and help coach colleagues in bug report writing.

How to Evaluate a Bug Report

Here are some common questions that I've found useful when evaluating bug reports. The value of the collection is that they help me quickly spot (and name) the weaknesses of reports that I read. I don't ask every question about every report, and if I was writing an evaluation, I would only highlight those answers that seemed the most insight-producing.

I do ask at least one question within each of the four categories:

1. What are my **first impressions** of the report?
2. What happens when I attempt to **replicate the report**?
3. What **follow-up tests** should have been done in the course of writing this report?
4. Does the report include **speculation or evaluation** by the tester, and if so, is it appropriate and useful?

Skim through this list as you read the report—don't work through every question. Your evaluation should point out the strengths of what you have read as well as the weaknesses.

Evaluation: First impressions

- Is there a summary?
    - Is it short (about 50-70 characters) and descriptive?
- Can you understand the report?
    - As you read the description, do you understand what the reporter did?
    - Can you envision what the program did in response?
    - Do you understand what the failure was?
- Is it obvious where to start (what state to bring the program to) to replicate the bug?
- Is it obvious what files to use (if any)? Is it obvious what you would type?
- Is the replication sequence provided as a numbered set of steps, which tell you exactly what to do and, when useful, what you will see?
- Does the report include unnecessary information, personal opinions or anecdotes that seem out of place?
- Is the tone of the report insulting? Are any words in the report potentially insulting?
- Does the report seem too long? Too short? Does it seem to have a lot of unnecessary steps? (This is your first impression—you might be mistaken. After all, you haven't replicated it yet. But does it LOOK like there's a lot of excess in the report?)
- Does the report seem overly general ("Insert a file and you will see" – what file? What kind of file? Is there an example, like "Insert a file like blah.foo or blah2.fee"?)

Evaluation: Replicate the Report

- Can you replicate the bug?
- Did you need additional information or steps?
- Did you have to guess about what to do next?

- Did you get lost or wonder whether you had done a step correctly? Would additional feedback (like, "the program will respond like this...") have helped?

- Did you have to change your configuration or environment in any way that wasn't specified in the report?

- Did some steps appear unnecessary? Were they unnecessary?

- Did the description accurately describe the failure?

- Did the summary accurate describe the failure?

- Does the description include non-factual information (such as the tester's guesses about the underlying fault) and if so, does this information seem credible and useful or not?

Evaluation: Follow-Up Tests

- Are there follow-up tests that you would run on this report if you had the time?

  - *In follow-up testing, we vary a test that yielded a less-than-spectacular failure. We vary the operation, data, or environment, asking whether the underlying fault can yield a more serious failure or a failure under a broader range of circumstances.*

  - *You will probably NOT have time to run many follow-up tests yourself. For evaluation, my question is not what the results of these tests were. Rather it is, what follow-up tests should have been run—and then, what tests were run?*

- What would you hope to learn from these tests?

- How important would these tests be?

- Are some tests so obviously likely to yield important information that you feel a competent reporter would have run them *and described the results?*

  - The report describes a corner case without apparently having checked non-extreme values.

  - Or the report relies on other specific values, with no indication about whether the program just fails on those or on anything in the same class (what is the class?)

  - Or the report is so general that you doubt that it is accurate ("Insert any file at this point" – *really? Any file? Any type of file? Any size? Did the tester supply* reasons for you to believe this generalization is credible? Or examples of files that actually yielded the failure?)

Evaluation: Tester's Speculation or Evaluation

Some bug reports include evaluative (rather than factual) remarks from the tester, such as hypotheses about the underlying fault or about the importance of the problem to the customer. The report need not include such information, but if it does, it should be credible, accurate, and useful.

- If the report includes such information, is it credible and useful or not?

- Does the report cite facts to support the evaluation?

- Is the evaluation based on special knowledge of the tester that might not be readily available to the programmer fixing the bug?

Phase 2

- **After you submit your comments, please peer review comments from two other participants in the course.**

When I grade bug evaluations, I generally do the following:

- Find the student's bug in the database
- Replicate it
- Consider the student's comments within the bug report (the student was supposed to strengthen the report)
- Comment (to the student--in this course, on the forum), using the core criterion is whether the student's comments would help the dev team assess and troubleshoot the bug
- Review the student's comments to our class about the bug report
- The core criterion is whether the student's comments show insight into good communication and troubleshooting
- In making comments, tie them to the assignment itself, with its multi-page list of criteria under which we evaluate bug reports

In my experience, the process of evaluating other students' evaluations of bugs leads many of my students to new insights. This is an important part of the process.

Sample Evaluation and Peer Review

Part 1: Here is an original bug report from the Open Office database--*This is the input to Phase 1.*
Part 2: Following that, I include a student's evaluation of the report. *This was an output from Phase 1.*
Part 3: Given the bug report and the student's evaluation, I can do the Phase 2 peer review. My review illustrates how I would have done the Phase 2 evaluation, along with my comments on what the student did.

Issue #:	*<number deleted>*	**Platform:**	PC		**Reporter:**	*<name deleted>*
Component:	Presentation	**OS:**	Windows 2000		**Add CC:**	
Subcomponent:	viewing	**Version:**	OOo 3.0 Beta			
Status:	UNCONFIRMED	**Priority:**	P3		**CC:**	None defined
Resolution:		**Issue type:**	DEFECT			
Assigned to:	cgu (cgu)					
QA Contact:	issues@graphics					
* Summary:	try to run slide show, and impress crashes.					

Attachments:	Date/filename:	Description:	Submitted by:
	Fri May 23 19:36:00 +0000 2008: test.ppt	ppt presentation created in 3.0 (application/vnd.ms-powerpoint)	*<name deleted>*
	Sat May 31 00:03:00 +0000 2008: trace.txt	Trace back from slide show crashing (text/plain)	*<name deleted>*

Description: Opened: Sat May 17 03:51:00 +0000 2008   Sort by: Oldest first | Newest first

---

```
Create slide show (3 slides!), try run slide show, and impress crashes. Happened
three times. Sufficient processor speed/RAM on my laptop.
```

------- Additional comments from <name deleted> Mon May 19 07:22:25 +0000 2008 -------

```
Reassigned.
```

------- Additional comments from <name deleted> Tue May 20 08:54:14 +0000 2008 -------

```
Do you use empty slides?
If this occurs with a specific document please attach the document.
```

------- Additional comments from <name deleted> Fri May 23 19:36:45 +0000 2008 -------

Created an attachment (id=53892)

```
ppt presentation created in 3.0
```

------- Additional comments from <name deleted> Sat May 31 00:03:57 +0000 2008 -------

Created an attachment (id=54103)

```
Trace back from slide show crashing
```

------- Additional comments from <name deleted> Sat May 31 00:04:33 +0000 2008 -------

```
Could be a dup of this issue:
```
http://qa.openoffice.org/issues/show_bug.cgi?id=90085
```
attaching stack trace.
```

------- Additional comments from <name deleted> Sat May 31 00:04:54 +0000 2008 -------

```
Could be a dup of this issue:
```

```
http://qa.openoffice.org/issues/show_bug.cgi?id=90085
```
attaching stack trace.

------- Additional comments from <BBST student> Sun Jun 1 03:03:39 +0000 2008 -------

This happens with the sample presentation at
```
http://www.sunvirtuallab.com:8001/sample_docs/109171/generic_fr.ppt
```
(winxp, beta3.0)

All other views are fine, but slide show crashes, though I did not get a
traceback to compare to the other issue.

By any use of this Website, you agree to be bound by these Policies and Terms of Use

==================================

2. The student's evaluation of this report

In a general sense, I agree with several other folks - you certainly see a broad spectrum of bug reports in their system. A lot of them don't provide a lot of detail - mostly because it seems that for the posters, the problem is persistent and easy to reproduce, so they assume it should be for everyone. This is especially true of folks reporting crash bugs.

http://www.openoffice.org/issues/sho...g.cgi?id=89572

I commented on this issue just to provide the information that it is easy to reproduce using the sample file provided in a different bug report I viewed - that sample includes a few languages, and sound.

Overall, this bug report was weak from the start - saying you have sufficient RAM/Memory is not a quantitative statement - just give the actual RAM/Memory, along with your exact OS information. Similar with the slides - there was no info given as to the features that might be on the 3 slides he's got; this particular issue may be related to similar stuff I read with issues on sounds.

Like someone else, I attempted to reproduce this issue also:
http://www.openoffice.org/issues/sho...g.cgi?id=89701

And even following the extra info the user provided, I had no issues at all with the steps he'd provided.

One thing that surprised me was more folks not including the file they're working with - a lot of times issues can be specific to some property or preference set in that file; not sure if they're encouraged not to, however, but it would seem to me that would make things far easier for the QA team and other folks attempting to reproduce their issues.

3. Instructor's comments on the report

Phase 1 of the Assignment involved two subtasks:

1. Edit the report to improve it, and

2. Evaluate the quality of the report.

1. Instructor comments on "Edit the report to improve it"

Your edit does not state whether you were able to replicate the failure that was reported. In addition, it does not describe your system configuration. Because of this, I don't know how to tell whether your report is helpful or a distraction.

When I attempted to replicate this bug, I downloaded the bug reporter's file, was unable to replicate the crash, and noticed an oddity in the file (I can't move or stretch the graphic). I also downloaded the file you pointed to and did crash on that. Note that the two files differ in an important respect: the first page of the one you point to has an animation.

My comment to the OOo team was:
------- Additional comments from cemkaner Sun Jun 1 19:03:10 +0000 2008 -------
```
I do not replicate this failure. I downloaded the attached file, ran the slide
show and it terminated normally. My system is WinXP, 4gB ram, dell m6300
notebook. I did notice with these slides that when I select the graphic on any
of the slides (I tried all 3), I get the selection boxes but cannot stretch or
reposition the graphic. Similarly, when I go to notes view, I can select the
slide (get the boxes at each corner) but I cannot stretch or move it.

I do replicate a crash with
http://www.sunvirtuallab.com:8001/sample_docs/109171/generic_fr.ppt and
therefore I think these may be distinct failures.
```

I think your comment was appropriate but would have been better if you had included
- your configuration information

- whether you could replicate the bug

2. Instructor comments on "Evaluate the quality of the report."

The assignment (pages 3-5) suggests several ideas for evaluating the bug. You cover some of them. No one would, or should, take the time to work through all of them for any particular report. That said, here is a slightly more structured set of comments on Issue 89572 (with a few notes on your comments)

First impressions:
- The summary is very broad: crash on slide show. OOo won't crash on EVERY slide show (or even most of them) and so I think there is critical information missing that I would expect in a summary. Still, it identifies severity and will show up in a search, so it meets several of the OOo project expectations.

- The replication conditions are very poor. The initial post implies that any 3-slide file will cause a crash. That's not true. The follow-up adds a 3-slide file (3 copies of the same slide) and a traceback. I still don't have any idea what is special about this file, if anything.

Replicate the Report

- I was unable to replicate the problem as reported, with the file supplied.

- The replication steps are, in essence, open my file and run a slide show, then you crash. This is insufficient.

- I don't know his configuration and so I don't know if this is a config difference. He is Win2k whereas I am WinXP.

- It's not clear why he creates a ppt file instead of an impress file. It's not clear whehter he created or imported the ppt file. His graphic is odd in that I cannot move or edit it, I don't know the properties of the graphic; could that be part of the problem? How did he create this file?

Follow-Up Tests

- I tried creating a file in Office (to make a ppt) with graphics, then importing into OOo. There was no failure.

- I tried creating a file in OOo (new file, insert a jpg graphic about the same size as the one in the reporter's test file, duplicate that slide twice to get three identical slides), saving it as ppt, exiting OOo, then opening the ppt into OOo, then doing a slide show. That all worked.

- I tried opening a long ppt file into OOo (the 197-slide bug advocacy slide set), in ppt and pptx formats. The pptx format was incorrectly read by OOo but did not cause a crash. The ppt file opened OK and I went through the full slide show w/o incident.

- I tried repositioning the graphics in the reporter's file and various other tasks to figure out what the graphic was (background? no...) without success.

- I also replicated the crash with the other file that you mentioned,

http://www.sunvirtuallab.com:8001/sample_docs/109171/generic_fr.ppt

so I can get a failure on this one, but not on the first one.

Overall evaluation of the Report

This was a weak report. It's style was OK but its information was insufficient.

Phase 3

In the third phase, you will pair up with another participant. Pick one UNCONFIRMED report on IMPRESS each. These must be different reports and they should be different from the reports that class has already worked on.

In this phase, before you submit any comments to the OpenOffice database or to our forum:

- **send your comments to your partner and get his or her feedback first. Then improve your work as appropriate and after that,**

- **file comments designed to improve the report with the OpenOffice tracking system**

- **file comments designed to evaluate the report on our forum**

- **file comments on how the peer review helped (if it did) and whether you think it was worth it (and why)**

At least two days before the start of Phase 3, you should tell the instructor who your partner is. Otherwise, the instructor will assign your partner.

The main mistake people make in Phase 3 is to ignore their partner, doing (and submitting) their work on their own. The second most common mistake is to give your work to your partner so late that s/he can't do anything effective with it.

Peer review makes a big difference in quality, but you won't experience this if you don't give it a fair chance.

If you're taking this as an online course, we recommend that you set up at least two phone calls to work through the details of your reviews of each other's work. Making four phone calls, one per day, would not be unusual. To make free phone calls across continents, we recommend skype.

Submissions should be just like Phase 1

- useful comments added the original bug report

- an evaluation report to the forum

but better.

Phase 4

In the fourth phase, you will grade two of the reports submitted in Phase 3, using the grading guidelines that we provide.

The way we'll usually work this is to assign a report to you and one to your partner.

- You are responsible for both, but you will be the lead reviewer for the one submitted to you and the assistant reviewer for the other one.

- Both of you should read both reports, compare notes on them, and then you write one evaluation, your partner writes the other.

Your ultimate grade for this assignment will be subjective and holistic. That is, I suggest that you form an overall impression and base your grade on that.

The two tables below can help you develop your impression. Your instructor will post a copy of the tables online, so that you easily copy the tables into your evaluation posting and write your notes into them.

0. THE TASK--HERE'S WHAT HAS BEEN SUBMITTED TO YOU

- A bug report number, that points you to a bug in the OOo database
- A bug report in the OOo database that has the comments of one student (or a partnered pair of students, but I'll pretend it is just one student)
- An evaluation report that tells us what the student thinks of the report.

Your total grade will combine your assessment of the bug itself (how the student improvement) and your assessment of the student's insight into the bug reporting communication, as reflected in the evaluation report.

Note that in the tables below, the score can reach a total above 10. That's because there are many ways to do a good job. I don't rely mechanically on the total. For example, suppose that in the evaluation report, the student does a broad but mediocre job (mediocre in terms of the quality of comments). The point total might be a 9 or 10 (of the total possible 15 points), but you might feel that it is C or B level work. In a holistic evaluation, you use categories to organize your thinking but ultimately follow your summary impression, not the score. So you would assign a grade of C or B to the work, rather than A.

1. ASSESSMENT OF THE BUG ITSELF

Allow 10 points for the bug report itself. The critical criterion is the extent to which the student's comments on the bug should be helpful to the OOo team:

- For example, if the student is reporting a failure to replicate, what makes the failure persuasive/useful? Seems to me that there are at least 4 items:
  - clear statement that it is irreproducible
  - clear statement of the steps followed, especially if the original report is unclear. If the original is clear and precise, it is sufficient to say that the listed steps were followed
  - clear description of the variations attempted, that go beyond the steps.
  - clear description of the configuration tested. More configs are better.
- For example, if the student is responding to a design issue, what information is brought to bear to assess the value of the requested design change?
  - oracle used?
  - market data / books / documentation / whatever?
  - scenarios?

SCORE YOU ASSIGN AND YOUR NOTES	Comments on the bug in the Issue Tracker record	Points possible (& notes) (Max total is 10)
	Student's comments are disrespectful in tone: 0 on the assignment	Zero on the assignment
	A follow-up test or discussion suggests that the student doesn't understand the bug (doesn't understand what is being complained about, and that lack of understanding is the fault of the student and not the bug report)	Best possible grade on the report = C (6/10)
	Clearly reports a simpler or clearer set of replication steps	up to +5
	Clearly reports one or more good follow-up tests	up to +5
	Good argument / data regarding importance	up to +8
	Student's comments include configuration & build	+1
	If the bug is in fact not reproducible, provides good description of effort to reproduce the bug, credibly demonstrates that this bug is not present on this configuration	up to +5
	Report a failure to repro on an alternate configuration without noting the config difference	-1
	Report a failure to repro on an alternate configuration that was already dismissed as irrelevant (or known not to yield the bug)	-5
	Report a criticism / failure to repro / other info that was already reported in another comment (unless the redundancy is intended to add value). The key thing here is to not waste time of the programming team.	up to -5 for each redundancy
	OVERALL EVALUATION--MAKE A HOLISTIC JUDGMENT	A+ = 10, A=8, B=7,C=6, D=5

2. ASSESSMENT OF THE STUDENT'S EVALUATION

I've provided notes on how to do this assignment in my notes on Phase 3. I recommend that in your use of the framework for evaluation (the list of 35 questions) the student should have answered about 10 of the questions.

- *How well did they do?*
- *How much insight did they show?*

SCORE YOU ASSIGN AND NOTES	CATEGORIES IN THE STUDENT EVALUATION	POINTS POSSIBLE (& notes) (Max total is 10)
	First impressions	up to 3
	Replication	up to 3
	Follow-up tests	up to 3
	Student's speculation or evaluation	up to 3
	Your closing impression of the sophistication of the student's work	up to 3
	Your closing impression of the insightfulness of the student's work	
	Any additional notes on what was done well	
	Any additional notes on what was done poorly	
	Any additional notes on what was missing that should have been there	
	OVERALL EVALUATION--MAKE A HOLISTIC JUDGMENT	A+ = 10, A=8, B=7,C=6, D=5

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 ed.). New York: Longman.

Association for Software Testing (2006). Code of Ethics Retrieved August 8, 2008, from http://www.associationforsoftwaretesting.org/drupal/ethics.

Association for Software Testing (2007a). About BBST-Foundations: Course Description Retrieved August 5, 2008, from http://www.associationforsoftwaretesting.org/drupal/courses/foundations.

Association for Software Testing (2007b). AST Guiding Principles Retrieved July 4, 2008, from http://www.associationforsoftwaretesting.org/drupal/principles.

Association for Software Testing (2008a). Course Description: BBST-Bug Advocacy Retrieved August 5, 2008, from http://www.associationforsoftwaretesting.org/drupal/courses/bugadvocacy.

Association for Software Testing (2008b). Instructor Policies: Certification of Instructors Retrieved August 8, 2008, from http://www.associationforsoftwaretesting.org/drupal/courses/policies.

Austin, R. D. (1996). *Measuring and Managing Performance in Organizations*: Dorset House.

Bach, J. (1999). General Functionality and Stability Test Procedure for Microsoft Windows 2000 Application Certification Retrieved March 1, 2008, from http://www.satisfice.com/tools/procedure.pdf.

Barber, S. (2008). Latest Column -- Inspired by Taking AST's Bug Advocacy Class Retrieved August 8, 2008, from http://www.testingreflections.com/node/view/7142.

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 Edition ed.). San Francisco: Jossey-Bass.

Byregowda, S. (2008). I Cleared the AST BBST Foundation Class :-) Retrieved August 8, 2008, from http://testtotester.blogspot.com/2008/07/i-cleared-ast-bbst-foundation-class.html.

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.

Cusumano, M. A., & Selby, R. W. (1998). *Microsoft Secrets*. New York: Free Press.

Day, J. A., & Foley, J. (2005). Enhancing the classroom learning experience with Web lectures: A quasi-experiment. *GVU Technical Report GVU-05-30*. Retrieved from ftp://ftp.cc.gatech.edu/pub/gvu/tr/2005/05-30.pdf

Day, J. A., & Foley, J. (2006, April). *Evaluating Web Lectures: A Case Study from HCI.* Paper presented at the CHI '06 (Extended Abstracts on Human Factors in Computing Systems), Quebec, Canada. Retrieved January 4, 2007, from http://www3.cc.gatech.edu/grads/d/Jason.Day/documents/er703-day.pdf.

Day, J. A., Foley, J., Groeneweg, R., & Van Der Mast, C. (2004). Enhancing the classroom learning experience with Web lectures. *GVU Technical Report GVU-04-18*. Retrieved from http://www3.cc.gatech.edu/grads/d/Jason.Day/documents/Day_042004.pdf

Day, J. A., Foley, J., Groeneweg, R., & Van Der Mast, C. (2005). *Enhancing the classroom learning experience with Web lectures.* Paper presented at the International Conference of Computers in Education, Singapore. Retrieved January 4, 2007, from http://www3.cc.gatech.edu/grads/d/Jason.Day/documents/ICCE2005_Day_Short.pdf.

Day, J. A., & Foley, J. D. (2006). Evaluating a web lecture intervention in a human-computer interaction course. *IEEE Transactions on Education, 49*(4), 420-431. Retrieved December 31, 2006.

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.

Drake, D. (2008). The AST Retrieved August 8, 2008, from http://www.thetested.com/2008/06/27/the-ast/.

Fiedler, R. (2008a). BBST Instructor Support Site Retrieved August 6, 2008, from http://bbstinstructors.org/.

Fiedler, R. (2008b). BBST Instructors Course Retrieved August 7, 2008, from http://www.associationforsoftwaretesting.org/moodle/course/view.php?id=27.

Fiedler, R. (2008c). BBST Instructors Manual Retrieved August 8, 2008, from http://bbstinstructors.wik.is/Manual.

Fiedler, R. (2008d). BBST Instructors Wiki: Fieldstones Retrieved August 7, 2008, from http://bbstinstructors.wik.is/Fieldstones.

Firstman, A. (1983). A comparison of traditional and television lectures as a means of instruction in biology at a community college. (Publication no. ED230264). from ERIC:

Foley, J., & Day, J. (2004-2006). HCC Web Lectures. *HCC Education Digital Library* Retrieved January 4, 2007, from http://hcc.cc.gatech.edu/taxonomy/webLectures.php.

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.

Fry, J. (2007). AST's Online Testing Courses, and Power of Two Bugs in Excel 2007 Retrieved August 8, 2008, from http://testingjeff.wordpress.com/2007/09/24/asts-online-testing-courses-and-power-of-two-bugs-in-excel-2007/.

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

Gannod, G. C., Burge, J. E., & Helmick, M. T. (2008). *Using the inverted classroom to teach software engineering*. Paper presented at the International Conference on Software Engineering, Leipzig.

Gates, B. (1997). Remarks by Bill Gates of Microsoft Corporation Retrieved January 16, 2006, from www.oasis-open.org/cover/gates-gartnerXML.html.

General Motors Corp. v. Johnston, 592 1054 (Supreme Court of Alabama 1992).

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 from http://ejournals.ebsco.com/direct.asp?ArticleID=NLAW20N8B16TQKHDEECM

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 from http://research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report&id=197

Hoffman, D. (1998). *A taxonomy of test oracles*. Paper presented at the International Software Quality Week. Retrieved February 2, 2008, from http://www.softwarequalitymethods.com/Papers/OracleTax.pdf

Hoffman, D. (1999). Heuristic Test Oracles: The balance between exhaustive comparison and no comparison at all. *Software Testing & Quality Engineering, 1*(2), 28-32.

International Institute for Software Testing (2008). Education-Based Software Test Certifications Retrieved August 6, 2008, from http://www.testinginstitute.com/downloads/Certification_Brochure.pdf.

Irvine, M. (2007). AST Black Box Software Testing Foundations Course Retrieved August 8, 2008, from http://rk-irvine.net/wordpress/index.php/2007/12/06/ast-black-box-software-testing-foundations-course/.

Jenkins, C. (2003, November 11). The year ahead for viruses. *Australian IT* from http://australianit.news.com.au/articles/0,7204,7826216%5e15302%5e%5enbv%5e,00.html.

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

Kaner, C. (1995). Testing Computer Software: Course Videotapes. Seattle, WA: ST Labs.

Kaner, C. (1997). The impossibility of complete testing. *Software QA, 4*(4), 28 from http://kaner.com/pdfs/imposs.pdf

Kaner, C. (2002). Principles of Software Testing for Testers (Videotape of course, used to train trainers). *Rational User Conference*. Retrieved from https://www-304.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=course_description&courseCode=RT101

Kaner, C. (2003). *Assessment in the software testing course*. Paper presented at the Workshop on the Teaching of Software Testing (WTST). from http://www.kaner.com/pdfs/AssessmentTestingCourse.pdf

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 from http://www.kaner.com/pdfs/CartsBeforeHorses.pdf.

Kaner, C. (2007). Writing multiple choice test questions Retrieved August 5, 2008, from http://www.satisfice.com/kaner/?p=24.

Kaner, C. (2008). BBST: Evolving a course in black box software testing--a report to the BBST Project Advisory Board Retrieved August 5, 2008, from http://www.kaner.com/pdfs/BBSTwtst2008AdvisoryBoard.pdf.

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

Kaner, C., Falk, J., & Nguyen, H. Q. (1993). *Testing Computer Software* (2nd ed.): International Thomson Computer Press; Reprinted,1999, by John Wiley & Sons.

Kaner, C., & Fiedler, R. (2005a, November). *Blended learning: A software testing course makeover.* Paper presented at the 11th Sloan-C International Conference on Asynchronous Learning Networks, Orlando, FL. Retrieved January 16, 2006, from http://www.ce.ucf.edu/asp/aln/2005sessions/presentations/1129300092852.pdf.

Kaner, C., & Fiedler, R. (2005b, October). *Inside out: A computer science course gets a makeover.* Paper presented at the Association for Educational Communications & Technology International Conference, Orlando, FL from http://www.kaner.com/pdfs/kanerfiedleraectprint.pdf.

Kaner, C., & Fiedler, R. (2007a). Adaptation & Implementation of an Activity-Based Online or Hybrid Course in Software Testing: Proposal to the National Science Foundation. Retrieved July 11, 2007, from http://www.kaner.com/pdfs/CirculatingCCLI2007.pdf.

Kaner, C., & Fiedler, R. (2007b). The AST BBST Series: Some Norms, Expectations & Tips: How do Students Pass this Course? Retrieved August 5, 2008, from http://www.associationforsoftwaretesting.org/BBST/passcourse.html.

Kaner, C., & Pels, D. L. (1998). *Bad Software: What To Do When Software Fails*. New York: Wiley.

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.

Knowles, M. S., Holton, E., F., & Swanson, R. A. (2005). *The Adult Learner: The Definitive Classic in Adult Education and Human Resource Development* (6th ed.). Burlington, MA: Elsevier (Butterworth-Heinemann).

Langer, N. (2002). Enhancing adult learning in aging studies. *Educational Gerontology, 28*(10), 895-904 from http://ejournals.ebsco.com/direct.asp?ArticleID=6MCFLPF1CKTAJ766WUKQ

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

Leuning, E. (1999). Analysts unveil final predictions for New Year Retrieved January 16, 2006, from http://news.com.com/2100-1091-234963.html?legacy=cne.

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

Minasi, M. (1999). *The Software Conspiracy: Why Companies Put Out Faulty Software, How They Can Hurt You and What You Can Do About It*. New York: McGraw-Hill. Retrieved August 8, 2008, from http://www.softwareconspiracy.com/swconspbook.pdf.

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

National Defense Industrial Association (2006). Top Software Engineering Issues within Department of Defense and Defense Industry. Retrieved from www.ndia.org/Content/ContentGroups/Divisions1/Systems_Engineering/NDIATopSWIssues06%20Report.pdf

National Institute for Science & Technology (2002). *NIST Planning Report 02-3, The Economic Impacts of Inadequate Infrastructure for Software Testing* from http://www.nist.gov/director/prog-ofc/report02-3.pdf.

Northern Arizona University Office of Academic Assessment (undated). Using the online Student Assessment of Learning Gains (SALG) for improving instruction and learning Retrieved January 3, 2007, from http://www4.nau.edu/assessment/oaa/services/salg.htm.

Oberg, J. (1999). Why the Mars probe went off course. *IEEE Spectrum*, 34-39. Retrieved January 16, 2006, from http://www.jamesoberg.com/mars/loss.html.

Osman, B. (2008). AST and the BBST Foundations Course Retrieved August 8, 2008, from http://bjosman.wordpress.com/2008/03/03/ast-and-the-bbst-foundations-course/.

Perrold, L. (2007). My excuses..... and the BBST Retrieved August 8, 2008, from http://www.mtom.co.za/Blogs/tabid/7356/EntryID/38/Default.aspx.

Raymond, E. S. (2001). *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Revised & Expanded Ed. ed.): O'Reilly Media.

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.

Sattsangi, P., Rutledge, J., Cooper, S., & Fox, S. (2008). *The next generation of MERLOT learning activities: Pedagogy in action.* Paper presented at the 8th Annual MERLOT International Conference, Minneapolis, MN. Retrieved August 8, 2008, from http://conference.merlot.org/2008/Friday/Rutledge_J_Friday.ppt.

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 from http://java.cs.vt.edu/public/classes/communities/readings/Savery-Duffy-ConstructivePBL.pdf

Schwartz, D. L., & Bransford, J. D. (1998). A Time For Telling. *Cognition and Instruction, 16*(4), 475-522.

Science Education Resource Center (undated). Pedagogy in Action: the SERC Portal for Educators: Submit an Activity Retrieved August 7, 2008, from http://serc.carleton.edu/sp/submit_activity.html.

Slabodkin, G. (1998, July 13). Software glitches leave Navy smart ship dead in the water. *Government Computer News*. Retrieved January 16, 2006, from http://www.gcn.com/17_17/news/33727-1.html.

Soundararajan, P. (2007). Pradeep fails BBST & continues to seek the Holy Grail Retrieved August 8, 2008, from http://testertested.blogspot.com/2007/09/pradeep-fails-bbst-continues-to-seek.html.

Student Assessment of Learning Gains (1997). Retrieved January 3, 2007, from http://www.wcer.wisc.edu/salgains/instructor/default.asp.

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

Weinberg, G. M. (2005). *Weinberg on Writing: The Fieldstone Method*. New York: Dorset House.

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.

Zeichick, A. (2005, April 1, 2005). Quality is Hot, Hi-B Visas are Not. *SD Times: The Industry Newspaper for Software Development Managers,* 5 from http://www.sdtimes.com/article/story-20050401-04.html.

Zele, E. v., 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.