# Practice and Transfer of Learning in the Teaching of Software Testing

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

Sowmya Padmanabhan, M.Sc.
*Microsoft TV*
*SowmyaP@microsoft.com*

***Abstract***

*Many university classes and commercial training courses rely on classroom lecture and practice exercises to help students learn new skills. The thesis work described in this paper sought to create an optimal set of lecture notes and practice exercises to teach a software testing technique, domain testing. The new materials were successful at teaching students to consistently and meticulously execute the procedures they learned. To the surprise of the authors, the students were unable to apply their new skills to a more applied task that was similar to those practiced. The results of this study provide a striking example of the limitations inherent in traditional instruction. They motivated Kaner to reconceptualize his instructional practices toward helping students deepen their appreciation of the complex cognitive tasks of the craft rather than procedurally-focused practice of new skills.*

## 1. Introduction

A common model for mathematical instruction relies on lectures with worked examples and lots of homework that applies and extends material introduced in lecture. For many students, the exercises provide the most significant learning experience—so significant that students often supplement their homework with even more exercises [24, 29]. This might not be the most effective way to teach mathematics, but it is extremely common. The quality of the homework exercises and grading comments seem to play a significant role in student satisfaction and, perhaps, learning success.

Through the 1990's, Kaner frequently taught commercial short courses in software testing (about 110 3-day classes plus many 1-day conference tutorials), evaluated courses taught by other instructors and compared notes on instructional style with other instructors, training companies, and large clients. The typical course was lecture style with a small number of group activities or exercises. A few courses offered gentle exams at the end. One of the factors limiting commercial training is time—a three-day course allows almost no time for practice, reflection, application to software the trainee is working with on the job, or critical discussion.

Kaner has also discussed the instructional style of university-level software testing courses with several instructors and students, at conferences [e.g., 15, 17, 18, 20, 21], in email discussions, and at five NSF-funded Workshops on Teaching Software Testing held annually at Florida Tech. Several of these courses are similar to the commercial courses—broad, lecture-style surveys.

Lectures can convey the *lecturer's* enthusiasm, which improves student satisfaction [32] and provide memorable examples to help students learn complex concepts, tasks, or cultural norms [11, 13, 23]. However, lectures are less effective for teaching behavioral skills, promoting higher-level thinking, or changing attitudes or values [3]. In terms of Bloom's taxonomy [1, 4], lectures would be most appropriate for conveying factual and conceptual knowledge at the remembering and understanding levels. Students of testing do need to learn

20th Conference on Software Engineering Education & Training (CSEET'07)
0-7695-2893-7/07 $20.00 © 2007 IEEE

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.

## 2. Domain testing

Domain testing [6, 7, 25, 26, 30, 31] is a stratified sampling strategy for choosing a few test cases from the near infinity of candidates [14]. This widely described strategy has several names, such as equivalence partitioning and boundary analysis. According to Richard Craig, a well known teacher, "equivalence partitioning is … intuitively used by virtually every tester we've ever met" [9, p. 162].

The essence of domain testing is that we partition a domain into subdomains (*equivalence classes*) and then select *representatives* of each subdomain for our tests. Variations of domain testing differ in the types of variables considered, and the rationales behind partitioning and selection of the representatives. The distinctions are not critical for this paper, but we have previously provided a literature summary [16].

Domain testing seems easy to explain but students can *appear knowledgeable* and *think they are knowledgeable* long before they achieve competence. For example, consider this task:

> "An integer variable can take values in the range -999 and 999, the endpoints being inclusive. Develop a series of tests by performing equivalence class analysis and boundary value analysis on this variable."

Asked to present and justify their analysis, students in Kaner's classes have often made these errors:

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

These errors reflect problems in higher-order thinking about domain testing. In terms of Bloom's taxonomy: *Applying a standard procedure* requires Level 3 (Application) knowledge. *Figuring out (and explaining the choice of) boundary values*

*within a given equivalence class* is a Level 4 (Analysis) task. *Identifying risks and associated error-revealing classes* is Level 6 (Evaluation).

## 3. Method

We believed that we could incrementally improve the typical testing course, and hoped to *substantially* improve students' ability to apply key testing techniques, by giving students more worked examples of the use of these techniques, more practice exercises, and more feedback on their exercises.

We started with domain testing because it is widely used and we could readily imagine how to create a series of practice questions that progressed in difficulty and explored different aspects of the technique. The pool of exercises and examples created for this study was more extensive than we would expect an instructor to use in a classroom setting. However, if effective, it could be repackaged for self-study (like Schaum's study aids for mathematics) and supplemented with video lectures [19].

Padmanabhan created lectures slides, examples, exercises and examinations for an 18 classroom-hour course in domain testing. All of the materials are available in her 661-page M.Sc. thesis [27]. The instructional style was procedural [10, 12]:

> "I have used the procedural approach in the training material for domain testing. First, I present my learners with the concept of domain testing technique. Next , I describe the tasks involved in performing testing using this technique. Then I lay out the procedures for performing the individual tasks, starting from the first and going to the last in a sequential and procedural manner.
>
> "This approach has its advantages and disadvantages. The disadvantages are apparent when higher-order learning is concerned, which I came to realize when my learners' performance [transfer] tests were evaluated. The advantage is that you train the learners to a common baseline and there is a lot of uniformity in what they learned and how they apply the knowledge" [27, p. 42-43].

These materials were reviewed and polished by Sabrina Fay (a Master's student in both Education and Software Engineering) and Rebecca Fiedler (a doctoral student in Education). Padmanabhan also taught early portions of the course to several volunteers and then taught two 15-hour pilot versions of the course with paid student subjects.

Work with student subjects was approved by Florida Tech's Institutional Review Board. The approval and subject consent forms are included in the thesis [27].

The 23 experimental subjects (*learners*) were undergraduate or graduate students at Florida Institute of Technology who had not yet taken a testing course but had completed the prerequisites required for Florida Tech's introductory testing course (one semester of discrete mathematics and at least two semesters of programming).[1]

The class was taught in five replications through the summer of 2003, with four or five learners per replication (a complete 18-hour class). Replications 1-4 were identical. The fifth differed from the others by correcting an error in our treatment of all-pairs combination testing [8]. That difference led us to exclude one pre/post-test question from our analysis and to ignore the answers to part of the transfer posttest.

On Day 1, Padmanabhan distributed reference materials to the learners and lectured for 30 minutes on the basic concepts of domain testing. The learners then completed a 90-minute, 9-question pretest that required learners to analyze integers, floating point variables, strings, Booleans, combinations of independent variables, and descriptive problems that required the learner to identify the variable(s) before doing the boundary/equivalence class analysis. The introductory lecture established a common

[1] Over the 10 semesters that Kaner has taught the introductory testing course at Florida Tech, undergraduates and graduate students appear to have performed equivalently on final exams.

set of definitions, reducing a risk that the pretest could underestimate the actual knowledge of a learner who simply didn't recognize the terminology we were using.

The pretest, like all subsequent exercises and tests, was open book. We made the lecture slides, worked examples and descriptions of tactics for solving several types of problems available to learners throughout the course. We did not allow learners to take materials home to discourage students from spending out-of-classroom time studying because this would have introduced substantial uncontrolled variation into the course.

There were two versions of the pretest, Pretest A and Pretest B, which we attempted to match in issues addressed and difficulty.

Days 2, 3 and 4 included morning and afternoon sessions of 80 to 120 minutes. Lectures laid out a class of problem (working with different data types, combinations, and spotting dependencies among variables) and worked one or more examples, illustrating a procedure for handling this class. Learners then practiced with exercises. The flow of each session alternated between short lectures and exercise sessions.

On Day 5, learners completed two post-tests.

The first posttest was pen-and-paper based. Learners who wrote Pretest A on Day 1 wrote Pretest B on Day 5 and those who started with Pretest B wrote Pretest A.

The second posttest, presented this task:

"For the *Page Setup* function of Microsoft PowerPoint application, whose screenshot is provided below, develop a series of tests for this function by performing equivalence class and boundary value analysis on it."

This dialog includes seven testable items: a list of preset page sizes (such as "Letter"), the Width and Height of the page (which might be set via selection of a preset size or might be custom-set by the user), page orientation ("Portrait" or "Landscape") of slides, page orientation of handouts, the slide number assigned to the first slide, and the Help button.

The instructions then provide a checklist, that started as follows:

"Make sure you include the following in your analysis:

a. List of variables; indicate input or output, their dimensions and the data type they map to.
b. Equivalence class table(s) that shows the complete equivalence class and boundary value analysis for the feature under test."

The rest of the checklist was different for the first 18 learners and the last five. The answers of the first 18 were generally weak in their handling of multiple variables and possible relationships among variables and we weren't sure whether the problem was poor underlying knowledge or unclear wording of the question. When we identified an error in the instructional materials associated with all-pairs combination testing, we decided to add another group of 5 learners who received better instruction on all-pairs testing. We also revised their transfer test to clarify the multiple-variable-related instructions. Had this group performed substantially better, we would probably have extended the study to see if the improvement replicated across additional groups.

The exact instructions are available in Appendix N of Padmanabhan's thesis [27, pp. 594-596]. In essence, both sets instructed learners to identify relationships among variables and discuss the test-related implications of them.

For this test, students worked at a computer. They could refine their test ideas as they worked with the program. This is slightly more complex than the examples worked in class, but relatively straightforward for an industrial application. This is a test of *transfer of learning*, which involves applying knowledge to a different task or a different context (see Chapter 3 of [5] for an excellent discussion.)

Finally, on each day, learners completed an evaluation of that day of instruction, rating such factors as clarity of lectures and exercises, their confidence that they can solve exercises, and satisfaction with instructor feedback. The evaluations also solicited comments, asking such questions as, "What did you like best about today's session?" and "What was the least useful part of today's session?"

## 4. Results

Course evaluations were generally favorable, rating various aspects of the course "Good" or "Excellent." All learners were "likely" (10/23) or "very likely" (13/23) to recommend this training to someone else. On Day 1, learners rated their competence in domain testing as "Excellent" (1), "Good" (7), "Fair" (11) and "Poor" (4). On Day 5, the ratings were higher, "Excellent" (6), "Good" (16), "Fair" (1) and "Poor" (0).

Padmanabhan graded Pretests (Posttests) A and B. Not surprisingly, learners did much better after they completed the course (mean posttest grade was 91.4, standard deviation of 6.2) than at start (mean pretest grade was 34.6, standard deviation of 11.3). All 23 learners wrote better posttests than pretests (binomial test, $p < 0.001$).

The results of the transfer test were less favorable than we expected.

Kaner, Patrick McGee, and James Bach graded the transfer test. Bach graded only the answers of the final group of five. All three graders had over 15 years of industrial software development experience, with an emphasis on software testing. Bach is a well known consultant (see http://www.satisfice.com for an overview of his work). McGee had recently come back to university and was a doctoral student at Florida Tech.

The evaluation criterion that we used to assess the answers relied on the graders' experiences recruiting, training and supervising testers:

> "Compare the results from this Performance [transfer] Test to the results that you would expect from a tester who claimed to have a year's experience and who claimed to be good at domain testing."

With respect to the distinction between the first 18 learners and the later group of 5, McGee noted a few ways in which the later group appeared to handle combinations of variables and dependencies among variables better than the first 18, but he was still strongly critical of their performance, as were Kaner and Bach (who graded only the last group's work). Even though the clarification of the instructions might have had a small effect, the similarities among the groups dwarf the differences and so we will treat the 23 as a homogenous collection.

The graders reached the same conclusion:

> "Overall, I think that the students learned the procedures well, but didn't learn domain testing well." (Kaner)
>
> "Overall, I did not get the impression that any of these subjects understood the material well enough to apply it to a new situation. I believe that they mostly could apply these techniques to situations that were very similar to the examples they had been trained on." (McGee)
>
> "The consistency among the answer sheets for each student suggests that the students either collaborated with each other to produce their answers, or that they were instructed in a very specific formula of domain testing….I expect more insight, product knowledge, and imagination from a serious tester who had more than a few months of experience." (Bach)

Kaner, McGee and Bach provided 12, 4, and 13 pages of comments that led to these conclusions. Here are five examples:

- As Bach noted, all learners approached the task in the same order, identified the same variables, analyzed them in essentially the same way and presented the results in extremely similar tables. The similarities were so pronounced that Bach

and Kaner wondered whether learners collaborated on the test. The experiment anonymized participation, but in later terms, some students identified themselves to Kaner, giving him an opportunity to interview them. He asked directly whether they had seen or heard of anyone collaborating during posttests; every student interviewed said "No." The consistency, striking across everyone in five separate offerings of the course, appears to come from clear procedural instruction, including the coached examples that learners worked in class.

- In considering page height and width, every learner discovered that PowerPoint accepts large values but automatically changes them to 56 inches. Every learner treated 56 as a firm upper bound. No learner considered whether the program behavior (sets the max at 56) might be incorrect, even though all of them said in one way or another that boundary-of-input tests are designed to check whether the program incorrectly accepts a value that is too large or rejects a value that is too small. No learner checked whether anything was different if the units of measurement were changed from inches to picas or points or centimeters.
- In considering page height and width, every learner checked the (empirically determined) boundary of 31 characters as the longest input string accepted and checked handling of ASCII boundaries (for example, the ASCII code for 0 is 48, so test entry of the slash (/) character, code 47). The course taught character set and length of the input string as examples of additional dimensions to consider when analyzing an input field. However, rather than learning these as examples, these learners appeared to treat these as an exhaustive list of the dimensions of interest. Other aspects of these variables, such as the number of digits after the decimal, number of decimal points in the string, handling of leading zeros, or the number of edits made on the field, were suggested rarely or never.
- The then-current version of PowerPoint gave an error message for large pages:

  "The current page size exceeds the printable area of the paper in the printer. Click Fix to automatically fit the page to the paper. Click Cancel to return to the Page Setup dialog box. Click OK to continue with the current page size."

  This message suggests a new set of equivalence classes—page sizes that will fit the printer versus sizes that will not. Every learner should have seen this message while checking page size limits in the Page Setup dialog, but no one mentioned it or based a test on it. Kaner asked a few students about this after they identified themselves as participants in the experiment. Some remembered too few details of the experiment to remember whether they saw this dialog. Others did remember it but could not explain why they did not base any tests on it.
- Another example of a missed opportunity—missed by every learner—involved checking the effect changing the page size had on the slide. In fact, the slide is rescaled. For example, in the then-current version of PowerPoint, if a slide is resized from 8.5" wide and 11" tall to 56" wide and 1" tall, text on that slide becomes short and wide. Some of the distortions look very bad. It was also possible to trigger intermittent errors (including a crash) by repeatedly resizing pages that had text and graphics. Not one test checked what the slides actually looked like after dimensions had changed. The tests all stopped at the dialog.

In this experiment, the lectures pointed out that it was a good idea to check the effect on the program of the values you enter. The lectures distinguished between input variables, whose values are set directly by the tester, and output variables, whose values are the results of inputs from the tester. Exercises required learners to identify some variables as output variables but learners were not required to set a value in one

variable and then trace the impact of that setting on some other variable. In the transfer test, the learners apparently forgot the lecture (or never understood this aspect of it), remembering only what they did in the exercises.

## 5. Discussion

It would be easy enough to modify the course to add a few more tasks to the procedure. For example, if a variable will affect a display, check the display after entering a value for that variable. If we added this to our instructions, some of the problems identified above would no longer arise. However, this would probably reflect a broader procedure that can be memorized and applied rather than a better understanding of the problem. The next issue that wasn't on the finite-length procedure would be missed, just as display-consequence was missed this time.

Our conclusion is that this practice-heavy, procedural style of instruction can prepare students well for an examination that calls for essentially the same analyses and applications as were practiced. Study guides that provide students with practice on the types of questions to be asked on exams will probably help those students pass exams. (Perhaps that's why over 30 million Schaum's Outlines have been sold.)

However, as with mathematics instruction, stellar performance on examinations doesn't mean that students can transfer the knowledge beyond the classroom [2, 5, 28].

In retrospect, perhaps we shouldn't have been surprised by this result. But we're not alone. When Kaner has mentioned it at meetings of trainers, the reaction has been surprise. Nor is this limited to the commercial training sphere. When Kaner summarized this result at an NSF meeting for principal investigators working on Science Technology Engineering & Mathematics education, several other PI's seemed surprised and asked when the result would be published.

For us, this result and the broader transfer literature that it instantiates encouraged us to fundamentally rethink our teaching style. Kaner has restructured the testing course so that lectures are on web, students watch them before coming to class, and class time is used for discussion and activities [22].

However, the challenge is not merely to make time for *lots of* activity. The domain testing instruction in this study included extensive coached, in-class activity. The challenge is to develop that group of activities that can foster insight—a level of abstract understanding that can apply from situation to situation—rather than emphasizing detailed procedural understanding.

## 6. Acknowledgements

This work was partially supported by NSF Grant EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers" and by a grant from Rational Software (IBM). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the supporting organizations.

We thank James Bach, Sabrina Fay, Patrick McGee, and Rebecca Fiedler for their assistance with this project.

## 7. References

1. Anderson, L.W., Krathwohl, D.R., Airasian, P.W., Cruikshank, K.A., Mayer, R.A., Pintrich, P.R., Raths, J. and Wittrock, M.C. A Taxonomy for Learning, Teaching & Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Longman, New York, 2001
2. Barnett, S.M. and Ceci, S.J. When and where do we apply what we learn? A taxonomy for far transfer. Psychological Bulletin, 128 (4). 612-637

3. Bligh, D.A. What's the Use of Lectures? Jossey-Bass, San Francisco, 2000
4. Bloom, B.S. (ed.), Taxonomy of Educational Objectives: Book 1 Cognitive Domain. Longman, New York, 1956
5. Bransford, J.D., Brown, A.L. and Cocking, R.R. (eds.). How People Learn: Brain, Mind, Experience and School (Expanded Edition). National Academy Press, Washington, D.C., 2000, from http://www.nap.edu
6. Clarke, L.A. A system to generate test data and symbolically execute programs. IEEE Transactions on Software Engineering, 2. 208-215
7. Clarke, L.A., Hassel, J. and Richardson, D.J. A close look at domain testing. IEEE Transactions on Software Engineering, 2. 380-390
8. Cohen, D.M., Dalal, S.R., Parelius, J. and Patton, G.C. The Combinatorial Design Approach to Automatic Test Generation. IEEE Software, 13 (5), from http://www.argreenhouse.com/papers/gcp/AETGissre96.shtml
9. Craig, R.D. and Jaskiel, S.P. Systematic Software Testing. Artech House, Norwood, MA, 2002
10. Driscoll, M.P. Psychology of Learning for Instruction. Allyn & Bacon, Needham Heights, MA, 2000
11. Forsyth, D., R. The Professor's Guide to Teaching: Psychological Principles and Practices. American Psychological Association, Washington, D.C., 2003
12. Gagne, R.M. The Conditions of Learning and Theory of Instruction. Holt, Rinehart & Winston, New York, 1985
13. Hamer, L. A folkloristic approach to understanding teachers as storytellers. International Journal of Qualitative Studies in Education, 12 (4). 363-380
14. Kaner, C. The impossibility of complete testing. Software QA, 4 (4). 28, from http://kaner.com/pdfs/imposs.pdf
15. Kaner, C., Incorporating software testing in the software engineering curriculum. in ACM SIGSOFT 2004/Foundations of Software Engineering/Educator's Grant Program tutorials, (Newport Beach, CA, 2004). Retrieved January 16, 2006, from http://www.isr.uci.edu/FSE-12/tutorials.html
16. Kaner, C., Teaching domain testing: A status report. in Conference on Software Engineering Education & Training, (Norfolk, VA, 2004), IEEE Computer Society. Retrieved January 16, 2007, from http://www.kaner.com/pdfs/teaching_sw_testing.pdf
17. Kaner, C., Teaching software testing. in meeting of Finnish university teachers of software testing, at the Tampere University of Technology (Tampere, Finland, 2004), from http://www.kaner.com/pdfs/kanerTampereTeaching.pdf
18. Kaner, C., Teaching the software testing course (tutorial). in Conference on Software Engineering Education & Training, (Norfolk, VA, 2004). Retrieved January 16, 2006, from http://www.kaner.com/pdfs/teaching_sw_testing.pdf; http://www.testingeducation.org/k04/index.htm
19. Kaner, C. and Bach, J. Black Box Software Testing: A Video Course, Melbourne, FL, 2005. Retrieved January 17, 2007, from http://www.testingeducation.org/BBST/index.html
20. Kaner, C. and Bach, J., Black box software testing: Tutorial on test design. in Pacific Northwest Software Quality Conference, (Portland, OR, 2003), from http://www.kaner.com/pdfs/PNSQCbbDesign.pdf
21. Kaner, C. and Fay, S., Teaching the Software Testing Course (Faculty poster). in Conference of the ACM Special Interest Group on Computer Science Education (SIGCSE 2004), (Norfolk, VA, 2004). Retrieved January 16, 2006, from http://www.kaner.com/pdfs/bbstSIGCSE.pdf
22. Kaner, C. and Fiedler, R. Adaptation & Implementation of an Activity-Based Online or Hybrid Course in Software Testing: Proposal to the National Science Foundation, 2007. Retrieved July 11, 2007, from http://www.kaner.com/pdfs/CirculatingCCLI2007.pdf
23. Kaufman, J.C. and Bristol, A.S. When Allport met Freud: Using anecdotes in the teaching of Psychology. Teaching of Psychology, 28 (1). 44-46
24. Mendelson, E. and Ayres, F. Schaum's Outline of Calculus. McGraw-Hill, New York, 1999
25. Myers, G.J. The Art of Software Testing. Wiley, New York, 1979
26. Ostrand, T.J. and Balcer, M.J. The Category-Partition Method for Specifying and Generating Functional Tests. Communications of the ACM, 31 (6). 676-686
27. Padmanabhan, S. Domain Testing: Divide & Conquer Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL, 2004.
28. Rebello, N.S., Cui, L., Bennett, A.G., Zollman, D.A. and Ozimek, D.J. Transfer of Learning in Problem Solving in the Context of Mathematics & Physics. in Jonassen, D.H. ed. Learning to Solve Complex Scientific Problems, Lawrence Earlbaum, Mahwah, NJ, 2007
29. Sveshnikov, A.A. Problems in probability theory, mathematical statistics and theory of random functions. Saunders, Philadelphia, 1968
30. Weyuker, E.J. and Jeng, B. Analyzing Partition Testing Strategies. IEEE Transactions on Software Engineering, 17 (7). 703-711
31. White, L.J., Cohen, E.I., & Zeil, S.J. A domain strategy for computer program testing. in Radicchi, B.C.S. ed. Computer Program Testing, North Holland Publishing, Amsterdam, 1981, 103-112
32. Williams, R.G. and Ware, J.E. 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