

On the Design of Advanced Courses in Software Testing

Cem Kaner
WTST 2014

Advanced Courses

Typical CS curriculum: First year

- CS -1 & CS-2 (two semesters total, introducing a programming language, fundamental concepts of control structures, data types and structures, operations on numbers, strings and complex data, problem solving with computers, and the use of tools to create, debug and test programs)
- Discrete math (1 semester). Introducing the essential math of programming.

THEN WE HAVE THE INTERMEDIATE COURSES

- Second year students take courses in data structures, software engineering process, software testing, computer architecture, programming in a second language, etc.

THEN WE HAVE THE ADVANCED COURSES

- Third year and fourth year courses on lots of more specialized and/or more deep topics.

We see a similar progression in other fields.

Engineer in Training (Wikipedia)

Engineer in Training, or EIT, is a professional designation from the National Council of Examiners for Engineering and Surveying (NCEES) used in the United States (and other countries) to designate a person certified by the state as having completed two requirements:

Completed a minimum of three years of post-secondary school at an ABET-accredited engineering program, or related science curriculum approved by the Board – Many states allow for the substitution of several years of engineering experience in place of the engineering degree requirement.

Passed the NCEES eight hour Fundamentals of Engineering (FE) Examination

Once an individual has passed the exam the state board awards that person an Engineer in Training (EIT) or an Engineer Intern (EI) designation. EIT and EI are equivalent variations in nomenclature that vary from state to state. Receiving an EIT designation is one step along the path toward Professional Engineer (PE) licensure.

Engineer in Training (Texas)

The term "Engineer-in-Training" defines a person certified by the State of Texas as one who is a graduate of an engineering program or related science curriculum approved by the Board and who has passed the National Council of Examiners for Engineering and Surveying (NCEES) eight-hour Fundamentals of Engineering (FE) Examination.

Individuals who meet the above criteria are eligible to apply for Engineer-in-Training certification. The Engineer-in-Training certification does not entitle an individual to practice as a professional engineer.

http://engineers.texas.gov/lic_eit_exinfo.htm

Engineer in Training (California)

Engineer-in-Training (EIT) certification applicants must have

Three years or more of postsecondary (college-level) engineering education. [Note: EIT applicants in the last semester or quarter of their third year of engineering education will be considered qualified.]

OR

Three years or more of engineering-related work experience.

OR

A combination of postsecondary (college-level) engineering education and engineering-related work experience totaling three years.

AND

Previously passed the NCEES Fundamentals of Engineering exam.

http://www.bpelsg.ca.gov/applicants/eit_lsitapp.shtml

The BBST Curriculum

Learning objectives: BBST (3-course set)

- Understand key testing challenges that demand thoughtful tradeoffs by test designers and managers.
- Develop skills with several test techniques.
- Choose effective techniques for a given objective under your constraints.
- Improve the critical thinking and rapid learning skills that underlie good testing.
- Communicate your findings effectively.
- Work effectively online with remote collaborators.
- Plan investments (in documentation, tools, and process improvement) to meet your actual needs.
- Create work products that you can use in job interviews to demonstrate testing skill.

Learning objectives: Foundations

- **First course of the BBST series:**
 - How to succeed in online classes
 - Fundamental concepts and definitions
 - Fundamental challenges in software testing
- **Improve academic skills**
 - Improve students' precision in reading
 - Create clear, well-structured communication
 - Provide (and accept) effective peer reviews
 - Cope calmly and effectively with formative assessments (such as tests designed to help students learn).
- **Learn about testing**
 - *Key challenges of testing*
 - Information objectives drive the testing mission and strategy
 - Oracles are heuristic
 - Coverage is multidimensional
 - Complete testing is impossible
 - Measurement is important, but hard
 - *Introduce you to:*
 - Basic vocabulary of the field
 - Basic facts of data storage and manipulation in computing
 - Diversity of viewpoints
 - Viewpoints drive vocabulary

Activities: Foundations

Orientation Puzzles / Discussions	Applications
1. What is the role of your testing group	1. Mission of Testing (What is the mission of the test group. Consider a comparable problem in 5 ways, in 5 hypothetical contexts. All students in all contexts.)
2. What oracles would you use in testing font display in a word processor? In particular, what oracles would support automation of the testing?	2. Apply the oracle-consistency heuristics to discovered bugs in 3 types of products. Which heuristics will be most/least effective? Why? What research would they lead you to do?
3. How would you test an integer square root function? (scope/size of the problem; risk tradeoffs)	
4. What are the characteristics of a valid metric? (evaluate a hypothetical bad one).	
5. Quiz after each of the first 5 lectures, with corrective feedback	
6. Post/discuss answers to sample exam	Final exam: Study-guide-based essays

These are all discussions of hypothetical situations, with different levels of depth and feedback.

Learning objectives: Bug Advocacy

Bug reports are not just neutral technical reports. They are persuasive documents. The key goal of the bug report author is to provide high-quality information, well written, to help stakeholders make wise decisions about which bugs to fix.

Key aspects of the content of this course include:

- Defining key concepts (such as software error, quality, and the bug processing workflow)
- The scope of bug reporting (what to report as bugs, and what information to include)
- Bug reporting as persuasive writing
- Bug investigation to discover harsher failures and simpler replication conditions
- Excuses and reasons for not fixing bugs
- Making bugs reproducible

- Lessons from the psychology of decision-making: bug-handling as a multiple-decision process dominated by heuristics and biases.
- Style and structure of well-written reports

Our learning objectives include this content, plus improving your abilities / skills to:

- Evaluate bug reports written by others
- Revise / strengthen reports written by others
- Write more persuasively (considering the interests and concerns of your audience)
- Participate effectively in distributed, multinational workgroup projects that are slightly more complex than the one in Foundations

Activities: Bug Advocacy

Orientation Puzzles / Discussions	Applications
1. Discuss: What is a bug / what is quality	1a. Replicate / improve / evaluate one unconfirmed bug in the OOO database 1b. Interactive grading of your work
2. Join the OpenOffice project, read their bug reporting standards	2. Peer-review Task 1 for two students
3. Discuss your experiences with bug reporting (extended questionnaire)	3. Redo Task 1 with another bug and more demanding assessment standards
4. Write a one-line summary of a bug based on a demonstration (series of screen shots)	4. Redo Task 2 (peer review) for one student, with more demanding assessment standards
5. (Post-lecture) Apply signal detection theory to management of test group's biases when deciding what bugs to report	
6. Quiz after each lecture, with corrective feedback	
Ongoing discussion of study guide questions	Final exam: Study-guide-based essays

*The applications involve deliberate practice with extensive feedback
and use a schema for evaluating the quality of bug reports*

Learning objectives: Test Design

This is an introductory survey of test design.

The course introduces students to:

- Many (nearly 200) test techniques at a superficial level (what the technique is).
- A detailed-level of familiarity with a few techniques:
 - function testing
 - testing tours
 - risk-based testing
 - specification-based testing
 - scenario testing
- domain testing
- combination testing.
- Ways to compare strengths of different techniques and select complementary techniques to form an effective testing strategy
- Using the Heuristic Test Strategy Model for specification analysis and risk analysis
- Using concept mapping tools for test planning.

Activities: Test Design

Orientation Puzzles / Discussions	Applications
1. (Post-lecture on risk) Create a list of input-variable related risks for a variable in OOo Writer	1. Use the HTSM to analyze the GoogleDrive specification for their presentation program
2. (Post-lecture on comparative strengths of test techniques) Compare / contrast 3 test techniques using 18-dimensional model	2a. Create a risk-equivalence table (combine risk analysis and equivalence class analysis) for an input variable 2b. Interactive grading of this task
3. (Post-lecture) Create classical boundary/equivalence tables for increasingly complex variables	
4. Apply the NIST ACTS tool and/or the MS PICT tool to plan a combination (configuration) test	
5. Quiz after each lecture, with corrective feedback	
Ongoing discussion of study guide questions	Final exam: Study-guide-based essays

The second application builds on a feedback'd lab (task-feedback-task) with detailed evaluation against a standard of professional-quality work

Learning objectives: Domain Testing

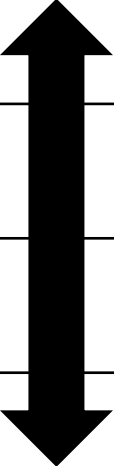
- Goal is to help students develop professional-practice level skill in application of this one technique.
 - Schema for domain testing
 - Exemplars of practice
 - Theoretical background material as needed to support understanding of the schema and the examples

Activities: Domain Testing

Orientation Puzzles / Discussions	Applications

*The applications involve deliberate practice with extensive feedback.
All activities and readings are organized around the Domain Testing Schema.
All activities involve test design, running through different parts of the Schema.
Final activity is a capstone project, with interactive grading.*

An example of a curriculum

	Foundations	Bug advocacy	Test design	Domain testing	Spec-based testing	Scenario-based testing
Greatest emphasis	Course skills	Testing skills	Testing knowledge	Testing skills	Testing skills	Testing skills
	Testing knowledge	Testing knowledge	Learning skills	Testing knowledge	Learning skills	Social skills
	Social skills	Social skills	Testing skills	Learning skills	Testing knowledge	Learning skills
	Computing fundamentals	Learning skills	Course skills	Computing fundamentals	Social skills	Testing Knowledge
	Learning skills	Course skills	Social skills	Social skills	Course skills	Computing fundamentals
Least emphasis	Testing skills	Computing fundamentals	Computing fundamentals	Course skills	Computing fundamentals	Course skills

Slide definitions

- *Course Skills*: How to be an effective student. Working effectively in online courses. Taking tests. Managing your time.
- *Social Skills*: Working together in groups. Peer reviews. Using collaboration tools (e.g. wikis).
- *Learning Skills*: How to gather, understand, organize and be able to apply new information. Using lectures, slides, and readings effectively. Searching for supplementary information. Using these materials to form and defend your own opinion.
- *Testing Knowledge*: Definitions. Facts and concepts of testing. Structures for organizing testing knowledge.
- *Testing Skills*: How to actually do things. Getting better (through practice and feedback) at actually doing them.
- *Computing Fundamentals*: Facts and concepts of computer science and computing-relevant discrete mathematics.

Which of these is “advanced”?

We can define “advanced” in terms of ...

- content
- institutional considerations
- skill development
- instructional style
- expectations of student performance
- credentialing

Bloom's taxonomy (updated by Anderson & Krathwohl)

The Knowledge Dimension	The Cognitive Process Dimension					
	<i>Remember</i>	<i>Understand</i>	<i>Apply</i>	<i>Analyze</i>	<i>Evaluate</i>	<i>Create</i>
<i>Facts</i>						
<i>Concepts</i>						
<i>Procedures</i>						
<i>Cognitive strategies</i>						
<i>Models</i>						
<i>Skills</i>						
<i>Attitudes</i>						
<i>Metacognition</i>						
Anderson Krathwohl model modified for software testing						

A course is advanced because it focuses on advanced content

In terms of the definition of “advanced”, I think the primary agreement in the instructional community is that there is no agreement about the *substance* of advanced courses. A course can be called advanced if it builds on other courses. Under this institutional definition, the ordering of topics and skills (introductory to advanced) determines what is advanced, but that ordering is often determined by preference or politics rather than by principle.

I am NOT just talking here about fields whose curricula involve a lot of controversy. Let me give an example. I am currently teaching Applied Statistics (Computer Science 2410). This is parallel in prerequisites and difficulty to the Math department’s course on Mathematical Statistics (MTH 2400). When I started teaching this, I made several assumptions about what my students would know, based on years of experience with the (1970’s to 1990’s) Canadian curriculum. I assumed incorrectly that students would learn very early about the axioms underlying algebra—this was often taught as Math 100 (1st course in the university curriculum). Here, it seems common to find that material in 3rd year. I also assumed incorrectly that my students would be very experienced in the basics of proving theorems. Again mistaken, and to my shock, many CS students will graduate, having taken several required math courses, with minimal skills in formal logic or theorem proof. I’m uncomfortable with these choices (in the “somebody moved my cheese” sense of the word “uncomfortable”)—it doesn’t feel right, but I am confident that these students studied other topics instead, topics that I would consider 3rd-year or 4th-year. Even in math, curriculum design is fluid and topics that some of us consider foundational, others consider advanced.

In a field like ours (testing) that is far more encumbered with controversy, there is a strong argument for humility when discussing what is “foundational” and what is “advanced”.

A course is advanced because the institution says so

In my experience, one of the challenges in teaching advanced topics is that many students will sign up who lack basic knowledge and skills, or who expect to use this course as an opportunity to rehash what they learned in their basic course(s). This is a problem in commercial and university courses, but in my experience, it is much easier to manage in a university because of the strength and visibility of the institutional support.

To make space for advanced courses, institutions that designate a courses as *advanced* are likely to

- *state and enforce prerequisites* (courses that must be taken, or knowledge/skill that must be demonstrated before the student can enrol in the advanced course)
- *accept transfer credit* (a course can be designated as equivalent to one of the institution's courses and serve as a prerequisite for the advanced course)

The designation *sets expectations*. Typically, this gives instructors room to:

1. limit class time spent rehashing foundational material
2. address topics that go beyond the foundational material (whatever material this institution has designated as foundational)
3. tell students who do not know the foundational material (or who cannot apply it to the content of the advanced course) that it is their responsibility to catch up to the rest of the class, not the course's responsibility to slow down for them
4. demand an increased level of individual performance from the students (not just work products on harder topics, but better work products that the student produces with less handholding from the instructor)

Note clearly that in an institution like a university, the decisions about what is foundational, what is advanced, and what prerequisites are required for a particular course are made by groups of instructors, not by the administrators of the institution. This is an idealized model—it is common for institutional administrators to push back, encouraging instructors to minimize the number of prerequisites they demand for any particular course and encouraging instructors to take a broader view of equivalence when evaluating transfer credits. But at its core, the administration adopts structures that support the four benefits that I listed above (and probably others). I think this is the essence of what we mean by “protecting the standards” of the institution

A course is advanced because it develops skills

I think of a skill as a type of knowledge that you can apply (you use it, rather than describe it) and your application (your performance) improves with deliberate practice.

Students don't just learn content in courses. They learn how to learn, how to investigate and find/create new ideas or knowledge on their own, how to find and understand the technical material of their field, how to critically evaluate ideas and data, how to communicate what they know, how to work with other students, and so on. Every course teaches some of these to some degree. Some courses are focused on these learning skills.

Competent performance in a professional field involves skills that go beyond the learning skills. For example, skills we must often apply in software testing include:

- many test design techniques (domain testing, specification-based testing, etc.). Testers get better with these through a combination of theoretical instruction, practice, and critical feedback
- many operational tasks (setting up test systems, running tests, noting what happened)
- many advanced communication skills (writing that combines technical, persuasive and financial considerations)

Taxonomies like Bloom's make the distinction between memorizable knowledge and application (which I'm describing as *skill* here). Some courses, and some exams, are primarily about memorizable knowledge and some are primarily about application.

In general, in my own teaching, I think of courses that focus on memorizable knowledge as survey courses (broad and shallow). I think of survey courses as foundational rather than advanced.

Most survey courses involve some application. The student learns to apply some of the content. In many cases, the student can't understand the content without learning to apply it at least to simple cases. (In our field, I think domain testing–boundary and equivalence class analysis—is like this.) It seems to me that courses run on a continuum, how much emphasis on learning things you can remember and describe versus learning ways to apply knowledge more effectively. I think of a course that is primarily a survey course as a survey course, even if it includes *some* application.

A course is advanced because it applies an advanced instructional style

Lecture courses are probably the easiest to design and the easiest to sell. Commercial and university students seem to prefer courses that involve a high proportion of live lecture.

Lectures are effective for introducing students to a field. They introduce vocabulary (not that students remember much of it—they forget most of what they learn in lecture). They convey attitudes and introduce students to the culture of the field. They can give students the sense that this material is approachable and worth studying. And they entertain.

Lectures are poor vehicles for application of the material (there's little space for students to try things out, get feedback and try them again).

In my experience, they are usually also poor vehicles for critical thinking (evaluating the material). Some lecturers develop a style that demands critical thinking from the students (think of law schools) but I think this requires very strong cultural support. Students understand, in law school, that they will flunk out if they come to class unprepared and are unwilling or unable to present and defend ideas quickly, in response to questions that might come from a professor at any time. Lawyers view the ability to analyze, articulate and defend in real time as a core skill in their field and so this approach to teaching is considered appropriate. In other fields that don't prioritize oral argumentation so highly, a professor who relied on this teaching style and demanded high performance from every student, would be treated as unusual and perhaps inappropriate.

As students progress from basic to advanced, the core experiences they need to support further progress also change, from lecture to activities that require them to do more—more applications to increasingly complex tasks, more critical evaluation of what they are doing, what others are doing, and what they are being told to do or to accept as correct or wise. Fewer things are correct. More are better-for-these-reasons or better-for-these-purposes

A course is advanced because we expect advanced performance

More advanced courses demand that students take more responsibility for the quality of their work:

- The students expect, and tolerate, less specific instructions. If they don't understand the instructions, the students understand that it is their responsibility to ask for clarification or to do other research to fill in the blanks.
- The students don't expect (or know they are not likely to get) worked examples that they can model their answers from or rubrics (step-by-step evaluation guides) that they can use to structure their answers. These are examples of *scaffolding*, instructional support structures to help junior students accomplish new things. They are like the training wheels on bicycles. Eventually, students have to learn to ride without them. Not just how to ride down this street for these three blocks, but how to ride anywhere without them. Losing the scaffolding is painful for many students and some students protest emphatically that it is unfair to take these away. I think the trend in many universities has been to provide more scaffolding for longer. This cuts back on student appeals and seems to please accreditors (university evaluators) but I think this delays students' maturation in their field (and generally in their education).

One of the puzzles of commercial instruction is how to assess student performance. We often think of assessment in terms of passing or failing a course. However, assessment is more broadly important, for giving a student feedback on how well she knows the material or how well she does a task. There has been so much emphasis on high-stakes assessment (you pass or you fail) in academic instruction that many students don't understand the concept of formative assessment (assessment primarily done to give the student feedback in order to help the student learn). This is a big issue in university instruction too, but my experience is that commercial students are more likely to be upset and offended when they are given tough tasks and told they didn't perform well on them. My experience is that they will make more vehement demands for training wheels in the name of fairness, without being willing to accept the idea that they will learn more from harder and less-well-specified tasks.

Things are not so well specified at work. More advanced instruction prepares students more effectively for the uncertainties and demands of real life. I believe that preparation involves putting students into uncertain and demanding situations, helping them accept this as normal, and helping them learn to cope with situations like these more effectively.

A course is advanced because it leads to an advanced credential

'I don't know what you mean by "glory",' Alice said.

Humpty Dumpty smiled contemptuously. 'Of course you don't — till I tell you. I meant "there's a nice knock-down argument for you!"'

'But "glory" doesn't mean "a nice knock-down argument",' Alice objected.

'When I use a word,' Humpty Dumpty said, in rather a scornful tone, 'it means just what I choose it to mean — neither more nor less.'

'The question is,' said Alice, 'whether you **can** make words mean so many different things.'

'The question is,' said Humpty Dumpty, 'which is to be master — that's all.'

(From Alice and Wonderland)

Foundations	Bug advocacy	Test design	Domain testing
Broad survey of conceptual issues	Focused look at bug reporting. Includes technical content (troubleshooting in order to prepare good reports), communication-skills content, and management	Advanced survey of somewhat-difficult, somewhat-technical content, with significant evaluation (compare/contrast strength/weakness).	Focused look at one test technique. Goal is to bring tester to “professional-level” application of that technique. Organized around a unifying schema and a textbook
Activities are primarily discussion of concepts	Activities include discussions and service learning (join a real project, do real work, reflect on what is being done)	Activities include discussions, classifications, and realistic-but-relatively-simple applications	All activities are applications of the schema to different parts of the test design problem. Some activities span a range of difficulty. Closing activity combines these in a capstone.
	<ul style="list-style-type: none"> • Skill-development • Institutional (Foundations prerequisite) • Expectations (skilled contribution to project) 	<ul style="list-style-type: none"> • Difficult content • Institutional (prerequisite) • Institutional note: The first 3 combined are CSE 3411/5411. Domain testing would be ½ of an advanced course with CSE 5411 prerequisite 	<ul style="list-style-type: none"> • Skill development • Difficult content, detailed demonstrations in multiple contexts • Institutional (prerequisites) • Expectations (more complex tasks)