Patterns of activities, exercises and assignments

Workshop on Teaching Software Testing
January 31, 2009

Cem Kaner, J.D., Ph.D.
kaner@kaner.com
Professor of Software Engineering
Florida Institute of Technology

These notes are partially based on research that was supported by NSF Grants EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers" and CCLI-0717613 "Adaptation & Implementation of an Activity-Based Online or Hybrid Course in Software Testing." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Overview

- I design and teach courses that rely heavily on activities, exercises, and assignments for their instructional effectiveness
- My terminology today (rough distinctions):
  - Activity.
    - Task < 45 minutes. (If done in-class, probably done in groups with instructor coaching.) (If online, done alone but posted to forum with encouraged discussion.) Probably focused on a single lecture or reading or a well-focused benefit. Assessment (grading credit) is for participation
  - Homework
    - Short task (< 1.5 hours) done by an individual or very small group. Grading could be for participation only or for strength of the work
  - Exercise
    - tightly specified task(s), each one narrow in scope. Often involves drill (repetitive tasks intended to give practice). Can be activity or homework
  - Assignment
    - Longer homework (1-2 calendar weeks). Often a group project. Integrative (scope is multiple lectures within or across units or a difficult reading. Assessment is for quality of the deliverable

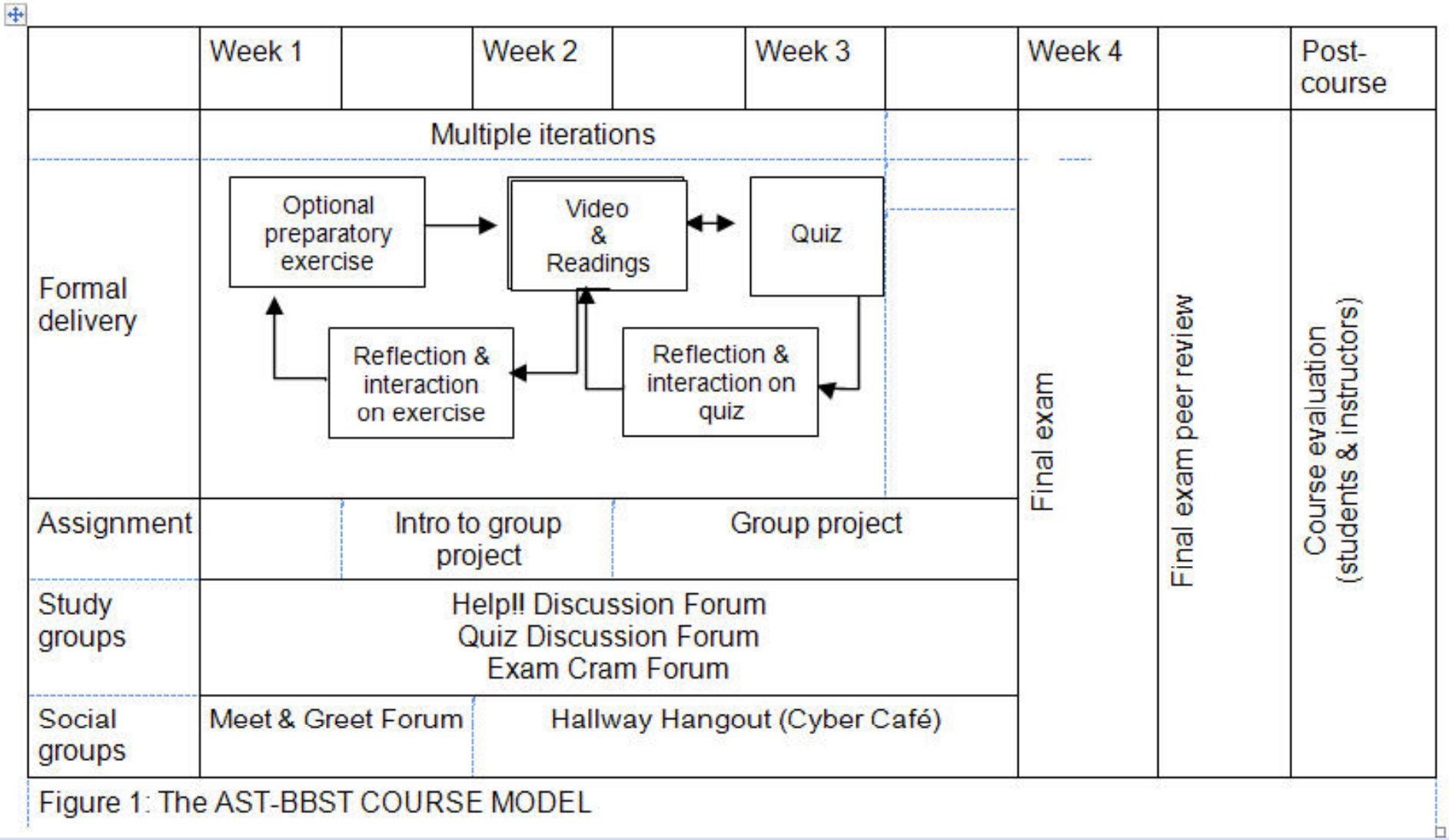

Figure 1: The AST-BBST COURSE MODEL

Discussion Questions

1. What are your three main objectives for in-class activities?
2. How do you assess whether an activity is effective?
3. Have you seen good collections of activities? If so, where would we find them? Have you tried any of them and, if so, how well did they work for you?
4. Have you seen any good classifications of activities or collections of patterns of activities?
5. Have you worked from generic activity descriptions to generate activities for your classes? How did that work?

Examples of Resources

NSDL Engineering Pathways

www.engineeringpathway.com/ep/catalog/long_catalog/

PR2OVE-IT

http://www.pr2ove-it.org/proveit/help/intervention.jhtml

WebQuest Taskonomy

http://webquest.sdsu.edu/taskonomy.html

WebQuest Design Patterns

http://webquest.sdsu.edu/designpatterns/all.htm

Activities Handbooks for the Teaching of Psychology

American Psychological Association (books)

Judi Harris' Virtual Architecture's Web Home

http://virtual-architecture.wm.edu/

SERC Pedagogy in Action Portal for Educators

http://serc.carleton.edu/sp/index.html

OnCore Blueprint catalog of 41 teaching repositories

http://www.oncoreblueprint.org/Repositories.htm

What are my activities' goals? (idiosyncratic, rough list)

Course-specific

- Preparatory task
  - advance organizers
  - expose the puzzle to be solved
- Understand content
- Check your comprehension
  - study guides / quizzes
- See implications or applications
- Generalize a concept
- Understand concept in context
- Relate to other concepts
  - compare/contrast
  - organize / classify
- **Generally, work with the material at any level of the Anderson / Krathwohl (Bloom's) taxonomy (next slide)**
- Develop a skill
- Do something valuable
- Prepare for exam

Broader skills

- Writing
- Oral presentation
- Statistical reasoning
- Empirical data collection
- Evaluate sources
- Critical reasoning
- Active reading
- Following instructions
- Precise reading
- Test-taking skills
- Peer review skills

In terms of content objectives, we might select any cell in the box

The Knowledge Dimension	*The Cognitive Process Dimension*					
	Remember	*Understand*	*Apply*	*Analyze*	*Evaluate*	*Create*
Factual knowledge						
Conceptual knowledge						
Procedural knowledge						
Metacognitive knowledge						

Original Anderson Krathwohl model

In terms of content objectives, we might select any cell in the box

The Knowledge Dimension	The Cognitive Process Dimension					
	Remember	**Understand**	**Apply**	**Analyze**	**Evaluate**	**Create**
Facts						
Concepts						
Procedures						
Cognitive strategies						
Models						
Skills						
Attitudes						
Metacognition						
http://www.satisfice.com/kaner/?p=14 *Anderson Krathwohl model modified for software testing*						

Task scope

Time available

- 30 minutes
  - e.g. preparatory exercise timebox
- class (lab) time
  - 50-75 minutes minus admin overhead
- long lab (3 hours)
- 2 days
- 1 week
- 2-3 weeks

How many people

- 1 (solo assignment)
- 1 but pairing allowed
- pairs
- small group
- interacting individuals
  - e.g. peer reviewers are interacting individuals, not a group
- interacting groups

- ***Must the instructor be present?***
- ***Synchronous or asynch?***

Example 1: Preparatory exercises

Objective of the **Oracle exercise**:
(http://www.testingeducation.org/k04/documents/pretest2004.pdf)

- Preparatory task--Help the student appreciate the complexity of the oracle problem so s/he can appreciate the lecture

Implementation

- Use a task that is familiar to the student and seems (misleadingly) to be superficially easy.
- Very few students can answer the questions well
- Time-box to 30 minutes b/c the goal is exposure,
- Brief peer review, help students see diversity of (wrong) approaches and (perhaps) the inadequacy of a solution they considered relevant
- Then lecture
- Then another peer review or debrief

Course-specific objective:

- Prepare for difficult lecture

Broader objective

- N/A

Time available:

- 30 min + 15 for peer review

How many people

- Interacting individuals (person plus peer review plus potential online discussion)

Example 1b: Scenario exercise

- Consider the use of Templates in OOo Impress.
- Why do people use them?
- What do people expect from them?
  - You might find it easiest to answer this by telling 1 to 3 short hypothetical stories that describe a hypothetical user, her expectations, and her attempt to use Templates to meet them.
- What makes a particular implementation of the template-related features good or bad?
  - You might find it easiest to answer this by telling 1 to 3 short hypothetical stories that describe a hypothetical user, her expectations, and her attempt to use Templates to meet them.
- These questions are often best addressed by scenario analysis, which we'll cover in the next lecture.
- In the meantime, please work with a partner and try to answer these questions yourself, making specific reference to the OOo templates and perhaps comparing them to some other product in the same market space.
- One thing to think about while you do this (I would welcome comments on this, too) -- how easy is it to come up with these hypothetical stories (scenarios) and how easy is it to explain your thinking using these stories, compared to explaining your thinking without these types of examples?
- Please submit your thoughts to the discussion forum.

Example 2: Preparatory: bug reporting

http://www.testingeducation.org/BBST/activities/ActivityBugReportingPaint.pdf

Objective:

- Preparatory task--Overcome resistance to the tedium of a lecture (and performance requirement) on bug reporting style / clarity

Implementation

- Have them write a bug report on a misleadingly simple-appearing bug
  - report results in a class forum or live discussion (list to the whiteboard) in class
  - instructor identifies common themes in the answers
- The assignment sets students up to make a uselessly vague bug report title
- VARIANT: add 45 minutes and have students write the actual bug report, then peer review for accuracy, clarity, and repeatability

Course-specific objective:

- Prepare for difficult lecture
- Skill development (apply lecture)

Broader objective

- Writing clarity

Time available:

- 30 min + 15-30 for review and discussion

How many people

- solo

Example 3: Preparatory: signup for bug reporting

http://www.testingeducation.org/BBST/activities/JoiningTheOpenOfficeProject.pdf

Objective:

- Preparatory task
- Several later parts of the course require finding bugs or reporting bugs.
- Join the Open Office project, sign up for the QA group, get privileges to report bugs, figure out how to search for bugs

Implementation

- Detailed instructions for signing up to the OOo site
- Instructor availability to answer tech-support and conceptual questions

Course-specific objective:

- Prepare (logistics) for assignment
- Familiarize with bug tracking system

Broader objective

- N/A

Time available:

- 30 min + 15-30 for review and discussion

How many people

- solo

Example 3: Do something useful

Objective:

- Apply complex lecture
- Integrate knowledge across several lectures

Implementation

- At this point, students have worked in detail with 3-4 test techniques and have gone through a survey of perhaps 8 more.
  - What makes test techniques different from each other?
  - Why use one and not another?

Let's look at the relevant parts of the lecture

Course-specific objectives:

- Integrate several lectures and prepare for deeper work on test design
- See relationships among test techniques

Broader objective

- Create a job-hunt-useful artifact

Time available:

- 1 week

How many people

- solo or small group

Software testing

- is an empirical
- technical
- investigation
- conducted to provide stakeholders
- with information
- about the quality
- of the product or service under test

We design and run tests in order to gain useful information about the product's quality

Testing is always a search for information

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

Different objectives require different testing tools and strategies and will yield different tests, different test documentation and different test results.

Test techniques

A test technique is essentially a recipe, or a model, that guides us in creating specific tests. Examples of common test techniques:

- Function testing
- Specification-based testing
- Domain testing
- Risk-based testing
- Scenario testing
- Regression testing
- Stress testing
- User testing
- All-pairs combination testing
- Data flow testing
- Build verification testing
- State-model based testing
- High volume automated testing
- Printer compatibility testing
- Testing to maximize statement and branch coverage

We pick the technique that provides the best set of attributes, given the information objective and the context.

Examples of test techniques

- **Scenario testing**
  - Tests are complex stories that capture how the program will be used in real-life situations.
- **Specification-based testing**
  - Check every claim made in the reference document (such as, a contract specification). Test to the extent that you have proved the claim true or false.
- **Risk-based testing**
  - A program is a collection of opportunities for things to go wrong. For each way that you can imagine the program failing, design tests to determine whether the program actually will fail in that way.

Techniques differ in how to define a good test

Power. When a problem exists, the test will reveal it

Valid. When the test reveals a problem, it is a genuine problem

Value. Reveals things your clients want to know about the product or project

Credible. Client will believe that people will do the things done in this test

Representative of events most likely to be encountered by the user

Non-redundant. This test represents a larger group that address the same risk

Motivating. Your client will want to fix the problem exposed by this test

Maintainable. Easy to revise in the face of product changes

Repeatable. Easy and inexpensive to reuse the test.

Performable. Can do the test as designed

Refutability: Designed to challenge basic or critical assumptions (e.g. your theory of the user's goals is all wrong)

Coverage. Part of a collection of tests that together address a class of issues

Easy to evaluate.

Supports troubleshooting. Provides useful information for the debugging programmer

Appropriately complex. As a program gets more stable, use more complex tests

Accountable. You can explain, justify, and prove you ran it

Cost. Includes time and effort, as well as direct costs

Opportunity Cost. Developing and performing this test prevents you from doing other work

Differences in emphasis on different test attributes

- **Scenario testing:**
- complex stories that capture how the program will be used in real-life situations
  - Good scenarios focus on validity, complexity, credibility, motivational effect
  - The scenario designer might care less about power, maintainability, coverage, reusability
- **Risk-based testing:**
- Imagine how the program could fail, and try to get it to fail that way
  - Good risk-based tests are powerful, valid, non-redundant, and aim at high-stakes issues (refutability)
  - The risk-based tester might not care as much about credibility, representativeness, performability—we can work on these after (if) a test exposes a bug

Software testing

- is an empirical
- technical
- investigation
- conducted to provide stakeholders
- with information
- about the quality
- of the product or service under test

There might be as many as 150 named techniques.
Different techniques are useful
to different degrees
in different contexts

Examples of important context factors

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

Example 3 continued: Do something useful

Implementation

- Create a chart: techniques (rows) and potential strengths (columns).
- For each technique, identify two fundamental strengths and one potential strength that is not inherent in this technique
- Write up a short explanation of your analysis of each technique
- Peer review and then create a second draft after the review, with notes (on a separate page) identifying changes made and why they are improvements

This is a powerful artifact to bring to a job interview

Course-specific objectives:

- Integrate several lectures and prepare for deeper work on test design
- See relationships among test techniques

Broader objective

- Create a job-hunt-useful artifact

Time available:

- 1 week

How many people

- solo or small group, peer reviews

Example 4: Complex, multi-phase skill development

http://www.testingeducation.org/BBST/assignments/AssignmentBugEvaluation.pdf

Objective:

- Develop skill in bug reporting
- Edit the work of others to learn quality standards for bug reporting, as a prerequisite to reporting bugs into a real-world project's (Open Office or Firefox) database

Implementation: 4-phase project

1. Find and evaluate report of unconfirmed bug,
   (a) improve the report for the project and
   (b) evaluate the report for the class
2. Peer review both outputs of Phase 1
3. Pair up, redo phase 1 but peer-review draft reports, then improve them before modifying original bug report or submitting evaluation
4. Peer review both outputs of Phase 3

Course-specific objectives:

- Apply lectures to real-life tasks
- Prepare for future tasks (report bugs)

Broader objective

- writing, write about same thing for different audiences, troubleshooting, peer review, following complex rubrics

Time available:

- 2 weeks

How many people

- interacting groups

Summing up my lecture:

At this point, my mental model is

- more focused on my objectives and my constraints
- not yet focused on activity patterns in an organized way

Discussion Questions

1. What are your three main objectives for in-class activities?
2. How do you assess whether an activity is effective?
3. Have you seen good collections of activities? If so, where would we find them? Have you tried any of them and, if so, how well did they work for you?
4. Have you seen any good classifications of activities or collections of patterns of activities?
5. Have you worked from generic activity descriptions to generate activities for your classes? How did that work?