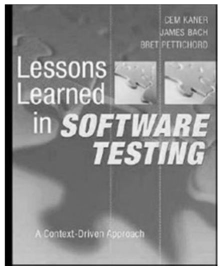
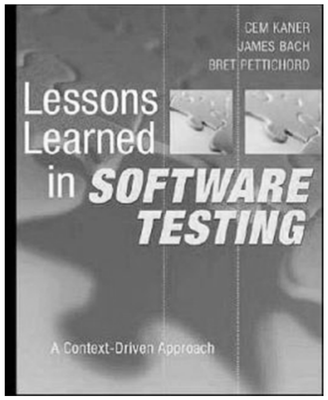

CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

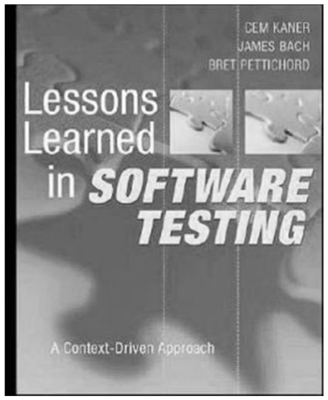
Theme 2: Thinking Like a Tester, Continued

Thinking Like a Tester


CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

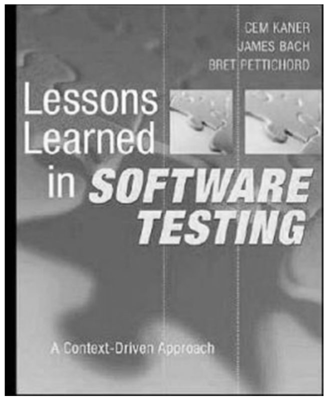
- Lesson 20: “Testing requires inference, not just comparison of output to expected results”
  - Must design tests and (usually) infer from a general spec the specific output that should result
  - There is no universal table from inputs->outputs
  - Must infer which other behaviors are “also tested” by each test

Thinking Like a Tester


CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

- Lesson 22: "Black box testing is not ignorance-based testing"
  - Requires *lots of knowledge:* of requirements, environment, configurations, data accesses, software this program works with, and of users
  - Just avoids building tests based on the source code as the primary source of test ideas
    - After all, to some extent the programmer can do that better, in unit tests

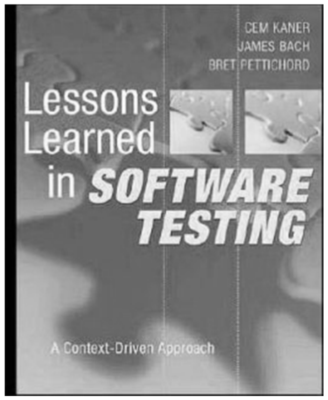
Thinking Like a Tester


CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

- Lesson 24: “All tests are an attempt to answer some question”

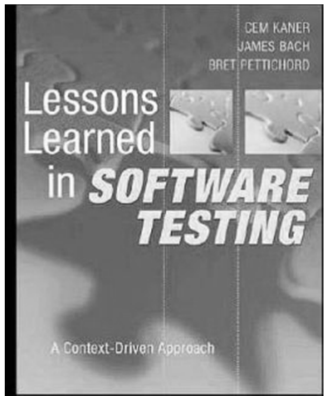
  - Epistemology again
  - If a test doesn’t answer any interesting questions, it probably isn’t worth running
    - Caveat: automatic generation may produce a huge number of tests where each in isolation is likely “useless” but the entire set of tests is highly effective
  - When manually testing, or scripting a specific test, think about (1) the question and (2) how to interpret the answer (are there ambiguous responses?)

Thinking Like a Tester


CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

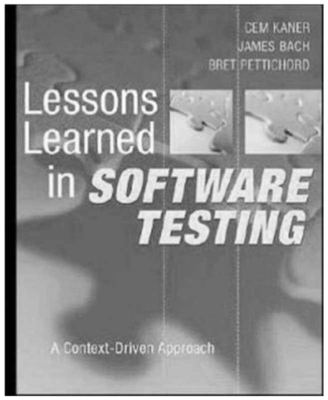
- Lesson 25: "All testing is based on models"
  - You have some model of what the software being tested should do
  - You may have an explicit reference model
  - You may model the various components and how they interact
  - You very likely will model the *user* of the software!

Thinking Like a Tester


CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

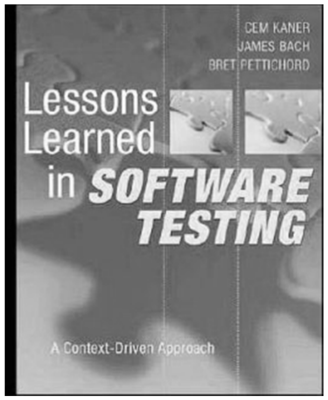
- Lesson 33: "Use implicit as well as explicit specifications"
  - Almost all software systems have (highly) incomplete specifications and requirements
  - You're going to have to decide what the software "should" do in cases that are not specifically addressed
  - Use common sense, general notions of software safety, security, reliability, and usability
  - Use *domain knowledge* of whatever the software is actually for

Thinking Like a Tester


CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

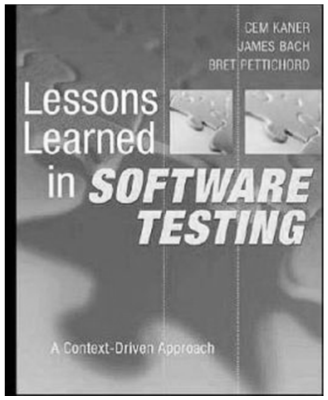
- Lesson 38: "Use heuristics to quickly generate ideas for tests"
  - Test boundaries & corner cases
  - Test error messages/conditions
    - "Rip out the hard drive, unplug the network"
  - Test unusual configurations
  - Run the tests that are annoying to run
  - Avoid redundancy, do not duplicate
    - For some critical conditions, automation's cheap and you're not sure it's totally redundant, try lots of small variations

Thinking Like a Tester


CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

- Lesson 41: "When you miss a bug, check whether the miss is surprising or just the natural outcome of your strategy"
  - Anyone can get unlucky
  - Make sure you aren't systematically going to "get unlucky" in this particular way

Thinking Like a Tester


CEM KANER
JAMES BACH
BRET PETTICHORD
Lessons Learned in SOFTWARE TESTING
A Context-Driven Approach

- Lesson 45: "… avoid '1287'"
  - This lesson I've left out part of (if you want the rest, read the book!)
  - Magic numbers are bad in test procedures & test code just as in normal programming
  - Explain the reasoning behind a "test procedure" (whether for a person to follow or a machine to run) like 'Type 1287 characters into field 1'
    - Why 1287?
    - For a human: "enter a very large (>1024) characters into the field"
    - For a machine: "fieldVal = genString(largeTextSize)"