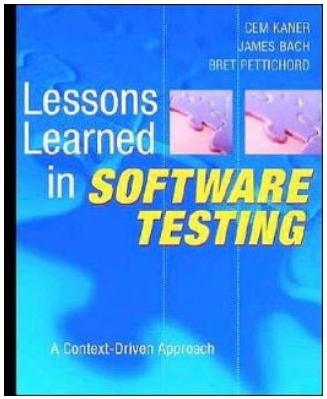
Topics for this Lecture



- Reporting bugs and ….

  working with others

Reporting Bugs

- **Lesson 55:** "*You are what you write*"
- Bug reports are the main "product" of testers
- **Bug reports::testers** as **source code::developers**
  - In heavily automated testing, your test code may also be a critical product, but it had better contribute to bug reports at some point
  - Managers and executives sometimes read bug reports.
  - You need to effectively make the case that this bug is worth giving up resources (money, programmer time, other development or bug fixing) to fix.

Reporting Bugs

- **Contents of a bug report (minimal)**
  - Unique ID (name/number).
    - People can find the bug again
    - There might be a bug database or a bug-tracking system, such as Bugzilla https://www.bugzilla.org/ or trac https://trac.edgewall.org/
  - What is the bug?
    - Symptom and Cause
  - How do you make the bug happen (BE SPECIFIC)?
    - If you have code that always produces the bug, include it!
    - If you can minimize (e.g., delta debugging; we'll talk about it later) do so
  - What version of the software was this detected on?
  - What is the estimated severity of the bug?
  - What is the estimated priority of the bug?

Reporting Bugs

- **Lesson 59:** "*Take the time to make your bug reports valuable*"
  - Bug reports are the main "product" of testers
  - Bug **reports::testers** as source **code::developers**
    - In heavily automated testing, your test code may also be a critical product, but it had better contribute to bug reports at some point
  - If your reports aren't understandable and informative, this is like producing bad, buggy code

Reporting Bugs

- **Lesson 68:** "*Never assume that an obvious bug has already been filed*"
  - Everyone may make this assumption…
    - And the bug will never get filed!

- **Lesson 71:** "*Uncorner your corner cases*"
- Programmers can sometimes ignore a test case that relies on particularly "odd" data:
  - You may try corner cases first since they are likely to fail
  - Once you find a bug, make sure you can't reproduce it with a simpler/less weird input
    - If you can, report that version instead!

Reporting Bugs

- **Lesson 73:** "Keep clear the difference between ***severity*** and ***priority***"
  - ***Severity*** is about the impact of a bug
    - *Severity* is about worst-case scenarios, probabilities, risks
    - **Examples** of high severity bugs: security compromises, incorrect results used in financial calculations, bugs that stop all testing and all the execution of the program.
- ***Priority*** is about how soon a bug should be fixed
  - Changes with time and circumstances
- *High severity* isn't always *high priority*:
  - If a bug corrupts any file saved in July 2010 only, it may not be important to fix
- *High priority* isn't always *high severity*:
  - Misspelling the company's name

Reporting Bugs

- **Lesson 82:** "Every bug deserves its own report"
- **Lesson 83:** "The summary line is the most important line in the bug report"
- **Lesson 86:** "Be careful of your tone. Every person you criticize will see the report"

Working with Others

- **Lesson 92:** "The best approach may be to demonstrate your bugs to the programmers"
  - As soon as they find a bug, some testers walk to the programmers who cover that area and describe it
  - Based on the discussion, they might do further troubleshooting or write up the report

Working with Others

- **Lesson 150:** "Understand how programmers think"
  - Programmers tend to specialize
    - They often focus on a subsystem or module
    - They often do not know the big picture very well
    - As a tester that may be your job
      - To test well, you have to understand how the system fits together
  - Programmers have a theory of the system
    - Report bugs in terms of programmers own models
  - Programmers often hate routine
    - They may think non-automated tests are "lame" or "wrong"

References:

C Kaner, J. Bach, B. Pettichord, Lessons Learned in Software Testing, Wiley, 2001.
http://classes.engr.oregonstate.edu/eecs/summer2015/cs362-002/Lecture11.pdf