CSE P503: Principles of Software Engineering

David Notkin
Spring 2009

Tonight's agenda

- Grades: 1st essay
  - https://catalysttools.washington.edu/gradebook/notkin/5243
- Software testing: general approaches, attitudes, and more
  - Also, time-permitting – Cooperative Bug Isolation and Test Prioritization
  - Next week: More technical stuff (concolic testing – mixing symbolic and concrete testing, etc.)
- Discussion: NATO and SWEBOK reports
- May 21st
- One-minute paper

Do you work in software testing?

- 30 seconds each to characterize what you do in testing…

Free association: "Software testing"

- Small groups then we'll merge

Groups of 3-4

Many points of view on testing

- Showing what you did is right
- Showing what somebody else did is wrong
- …more?

Steve McConnell

- "Testing by itself does not improve software quality. Test results are an indicator of quality, but in and of themselves, they don't improve it. Trying to improve software quality by increasing the amount of testing is like trying to lose weight by weighing yourself more often. What you eat before you step onto the scale determines how much you will weigh, and the software development techniques you use determine how many errors testing will find. If you want to lose weight, don't buy a new scale; change your diet. If you want to improve your software, don't test more; develop better."

Cem Kaner & James Bach

- "Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the software under test."
- "Testing is questioning a product in order to evaluate it.
  - "The 'questions' consist of ordinary questions about the idea or design of the product, or else questions implicit in the various ways of configuring and operating the product.
  - "The product 'answers' by exhibiting behavior, which the tester observes and evaluates."

Herb Simon (via wikipedia)

- "*Satisficing* … is a decision-making strategy which attempts to meet criteria for adequacy, rather than to identify an optimal solution. A satisficing strategy may often be (near) optimal if the costs of the decision-making process itself, such as the cost of obtaining complete information, are considered in the outcome calculus."
- "[Simon] pointed out that human beings lack the cognitive resources to maximize: we usually do not know the relevant probabilities of outcomes, we can rarely evaluate all outcomes with sufficient precision, and our memories are weak and unreliable. A more realistic approach to rationality takes into account these limitations: This is called bounded rationality."

Don Knuth

- "Beware of bugs in the above code; I have only proved it correct, not tried it."

Edsger Dijkstra

- "Program testing can be used to show the presence of bugs, but never to show their absence!"

Pradeep Soundarajan

- "It is not a test that finds a bug but it is a human that finds a bug and a test plays a role in helping the human find it."

A few more

- "Testing is a skill. While this may come as a surprise to some people it is a simple fact." (Fewster, Graham)
- "Testing a product is a learning process."(Marick)
- "Everything really interesting that happens in software projects eventually comes down to people." (Bach)
- "Any process that tries to reduce software development to a 'no brainer' will eventually produce just that: a product developed by people without brains." (Hunt, Thomas)

http://www.experiencefestival.com/

- "There is considerable controversy among testing writers and consultants about what constitutes responsible software testing. The self-declared members of the Context-Driven School of testing believe that there are no 'best practices' of testing, but rather that testing is a set of skills that allow the tester to select or invent testing practices to suit each unique situation. This belief directly contradicts standards such as the IEEE 829 test documentation standard, and organizations such as the FDA who promote them."

Top 10 Software Testing Quotes

http://www.jinsblog.com

- In God we trust, and for everything else we test.
- If it works, its the developer, if not it's QA
- Software Testers : We succeed where others fail!
- Software Testers Always go to Heaven ... they've already had their share of Hell!
- Only certainties in life: Death, taxes and bugs in code!
- Every morning is the dawn of a new error
- A bug in the hand is better than one as yet undetected.
- I don't make software; I make software better.
- The Definition of an Upgrade: Take old bugs out, put new ones in.
- All code is guilty, until proven innocent.

Standard testing questions (M. Young)

- Did this test execution succeed or fail?
  - Oracles
- How shall we select test cases?
  - Selection, generation
- How do we know when we've tested enough?
  - Adequacy
- What do we know when we're done?
  - Assessment?

Testing theory

- Plenty of negative results
  - Nothing guarantees correctness
  - Statistical confidence is prohibitively expensive
  - Being systematic may not improve fault detection (as compared to simple random testing)
- "So what did you expect, decision procedures for undecidable questions?"

What information can we exploit?

- Specifications: formal or informal
  - In oracles
  - For selection, generation, adequacy
- Designs …
- Code …
- Usage (historical or models)
- Organization's experience

17

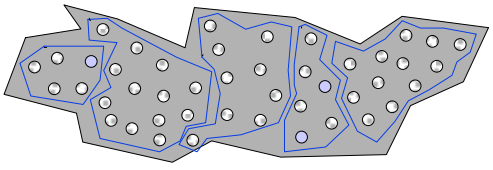
When can we stop?

- Ideally: adequate testing ensures some property (proof by cases)
  - Goodenough & Gerhart, Weyuker & Ostrand
  - In reality, as impractical as other program proofs
- Practical adequacy criteria are really "inadequacy" criteria
  - If no case from class X has been chosen, surely more testing is needed …

18

Partition testing

- Basic idea: divide program input space into (quasi-) equivalence classes, selecting at least one test case from each class
- The devil is in the details – and there are many!



19

Structural coverage testing

- (In)adequacy criteria – if significant parts of the program structure are not tested, testing is surely inadequate
- Control flow coverage criteria
  - Statement (node, basic block) coverage
  - Branch (edge) and condition coverage
  - Data flow (syntactic dependency) coverage
  - Others…
- "Attempted compromise between the impossible and the inadequate"

20

Statement coverage

- Unsatisfying in trivial cases

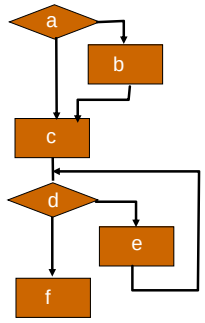
```
if x > y then
  max := x
else
  max :=y
endif

if x < 0 then
  x := -x
endif
z := x;
```

21

Edge coverage

- Covering all basic blocks (nodes, statements) would not require edge **ac** to be covered
- Edge coverage requires all control flow graph edges to be coverage by at least one test



22

Condition coverage

- How to handle compound conditions?
  - `if (p != NULL) && (p->left < p->right) …`
- Is this a single conditional in the CFG? How do you handle short-circuit conditionals?
- Condition coverage treats these as separate conditions and requires tests that handle all combinations
- Modified Condition/Decision Coverage (MCDC)
  - Sufficient tsest cases to verify whether every condition can affect the result of the control structure
  - Required for aviation software by RCTA/DO-178B

23

Path coverage

- Edge coverage is in some sense very static
- Edges can be covered without covering actual paths (sequences of edges) that the program may execute
- Note that not all paths in a program are always executable
  - Writing tests for these is hard ☺
  - Not shipping a program until these paths are executed does not provide a competitive advantage ☺

24

Path coverage

- The test suite **{<x = 0, z = 1>, <x = 1, z = 3>}** executes all edges, but…

```
if x ≠ 0 then
  y := 5;
else
  z := z - x;
endif;
if z > 1 then
  z := z / x;
else
  z := 0;
end
```

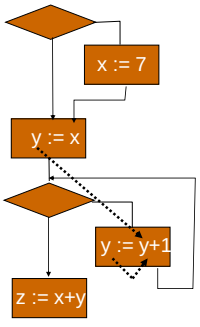
25

Loop coverage

- Loop coverage also makes path coverage complex
  - Each added iteration through a loop introduces a new path
  - Since we can't in general bound the number of loop iterations, we often partition the paths for testing purposes
    - Never, once, many times …
    - 10 is a constant often used as a representation of "many"

26

Data flow coverage criteria



- Idea: an untested def-use pair could hide an erroneous computation
- The increment of **y** has two reaching definitions
- The assignment to **z** has two reaching definitions for each of **x** and **y**
- There are many variants on this kind of approach

27

Structural coverage: challenges

- Interprocedural coverage
  - Interprocedural dataflow, call-graph coverage, etc.
- Regression testing
  - How to test version P' given that you've tested P
- Late binding in OO – coverage of polymorphism
- Infeasible behaviors: arises once you get past the most basic coverage criteria

28

Infeasibility problem

- Syntactically indicated behaviors that are not semantically possible
- Thus can't achieve "adequate" behavior of test suites
- Could
  - Manually justify each omission
  - Give adequacy "scores" – for example, 95% statement, 80% def-use, …
  - [Can be deceptive, of course]
- Fault-injection is another approach to infeasibility

Context driven testing: 7 Principles

http://www.context-driven-testing.com/

- The value of any practice depends on its context.
- There are good practices in context, but there are no best practices.
- People, working together, are the most important part of any project's context.
- Projects unfold over time in ways that are often not predictable.
- The product is a solution. If the problem isn't solved, the product doesn't work.
- Good software testing is a challenging intellectual process.
- Only through judgment and skill, exercised cooperatively throughout the entire project, are we able to do the right things at the right times to effectively test our products.

Principles in action: illustrations

- Testing groups exist to provide testing-related services. They do not run the development project; they serve the project.
- Testing is done on behalf of stakeholders in the service of developing, qualifying, debugging, investigating, or selling a product. Entirely different testing strategies could be appropriate for these different objectives.
- It is entirely proper for different test groups to have different missions. A core practice in the service of one mission might be irrelevant or counter-productive in the service of another.

Continued

- Metrics that are not valid are dangerous.
- The essential value of any test case lies in its ability to provide information (i.e. to reduce uncertainty).
- All oracles are fallible. Even if the product appears to pass your test, it might well have failed it in ways that you (or the automated test program) were not monitoring.
- Automated testing is not automatic manual testing: it's nonsensical to talk about automated tests as if they were automated human testing.
- Different types of defects will be revealed by different types of tests--tests should become more challenging or should focus on different risks as the program becomes more stable.
- Test artifacts are worthwhile to the degree that they satisfy their stakeholders' relevant requirements.

An example from Bach

- Asks students to "try long inputs" for a test requiring an integer
- Interesting lengths are…?

Key boundaries: most not tried

- 16 digits+: loss of mathematical precision
- 23+: can't see all of the input
- 310+: input not understood as a number
- 1000+: exponentially increasing freeze when navigating to the end of the field by pressing <END>
- 23,829+: all text in field turns white
- 2,400,000: reproducible crash

- Why more not tried?
  - Seduced by what's visible
  - Think they need the specification to tell them the maximum – and if they have one, stop there
  - Satisfied by first boundary
  - Use linear lengthening strategy
  - Think "no one would do that"

My view: testing has two objectives

- Identifying bugs
- Building confidence
  - More accurately, testing is one important dimension of building confidence in a software systems

SWEBOK: discussion

NATO 1968-69: discussion

Cooperative bug isolation (Liblit)

Test prioritization (Srivastava & Thiagarajan)

Optional…

- One-minute paper: Key point? Open question? Mid-course correction?