Defect testing

Goals:
– detect as many defects as possible
– detect the most damaging defects
– detect the most likely defects - statistical test!

**Black-box testing:** the source code is not considered (maybe even not known).

**Glass-box testing:** the tests are chosen based on the source code.

What is a test?

• A test suite is a set of test cases run together for a single purpose.

• A test case consists of
  – Test data
    • Including invalid inputs
  – Expected outcome (correct answer)
  – Expected behaviour (e.g. response time)

Black-box testing [8.1.2]

No code – but requirements!

**Partition testing:**
• Partitions: input and output equivalences.
  – **typical values**
  – **boundary values**
  – **invalid inputs**

Example: sorting a list

• length of list
  – empty list: boundary value
  – list with one element: boundary value
  – list with some "typical" number of elements
  – list with extremely many elements: boundary value

• comparisons
  – no duplicates (typical?)
  – some duplicates (typical)
  – all elements are the same (boundary value)

• invalid inputs
  – not a list
  – a list with elements that cannot be compared

Glass-box testing

"All code" should be tested at least once
– testing once is rather weak
– what does "all code" mean?

**Definition:** coverage is the percentage of "all code" that is tested by a test suite.

Coverage

• **Statement coverage**
  – every statement must be tested

• **Branch coverage**
  – every choice (if, while) must be tested for both true and false.
Example (coverage)

**Specification**
inputs: `result`, `taxrate`, `threshold`
output: `tax`
relation: `tax` is `<taxrate>` % of the profit, but the first `<threshold>` SEK is not taxed.
glossary: `profit` - a positive result.

Coverage testing flaws
Coverage testing tests code that exists, but
• not under all conditions,
• not code that should exist, but doesn’t.

How to do coverage testing?
• Decide on `paths` that cover the code
• For each path:
  – compute an input that will produce this path,
  – run a test with this input.
• Problems:
  – how to compute inputs for a given path,
  – there may be no such input (infeasible path)

How to do coverage testing? (theoretically)
• Decide on test data
  – based on … (statistical, partitioning)
• Use a testing tool that records
  – which code is executed during the test,
  – computes coverage.
• Problem:
  – you reach 80%, 90% or 95% coverage,
  – obscure code is only reached for very specific input
  – dead code is not executed for any input.
tax = 0
result > 0
Y

Y

N

N

tax = taxrate/100 * (result-threshold)
tax < 0

N
tax = 0

infeasible path

What input for this path?
taxrate/100 * (result-threshold) < 0

Y

tax < 0
N
tax = 0
backwards reasoning

Further coverage criteria

• Condition coverage
  % find item in array A
  i = 1;
  found = false;
  while i ≤ A.length and not found do
    if A[i] = item then ...
  end if;
  % oops
  tax = 0
  if tax < 0 then % oops
  tax = 0
  end if;

• requires tests
  i ≤ A.length and not found
  i > A.length
  found is true
  obvious in partition test!

• Data flow coverage
  for each variable, connect set – use pairs

• Relational operator coverage
  for each comparison a < b,
  test boundary cases
  – a = b
  – a = b-1

• Path coverage
  – every feasible path is covered
  – for programs without loops
  – 100% path coverage does not guarantee correctness!

Software Engineering V&V
• Loop coverage
  execute each loop
  – 0 times (if possible)
  – 1 time
  – several times

• Error message coverage
  – force the system to produce every error message

Interface coverage tests

• Function coverage
  – every function is called at least once (weak)

• Call coverage
  – every function call is executed at least once

Testing concurrent systems

• Problems:
  – What is a path? Sequence of executed statements from more than one source code.
  – A combinatorial explosion
  – Programmers make errors because of unforeseen sequences.
  – Hard to control which sequence is tested
    • errors may be hard to reproduce.
    • difference between “laboratory” and “reality”.

Summary

• Black box
  – Based on specification
  – “Natural”

• Glass box
  – Coverage ... Different criteria
    – 100% coverage ≠ 100% correct

• Concurrent systems
  – Testing may find some faults