Testing

Roland Bol
Test planning

- Determine goals
- Ensure testability
- Design tests
- Test tools?
- Time plan
- Evaluation criteria - what is good enough?
- Documentation
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
  - Expected outcome
  - Expected behaviour
Problems:
- Which input data should be tested?
- How do we know that the output is correct (oracle problem)?

Reusability:
- Regression test - rerun a test suite for every new version of the system
- Back-to-back testing - use a previous version of the system as the oracle
The oracle problem

What is the correct answer?
1. ... at least the program didn't crash ...
2. Compute by hand and compare
3. Back-to-back testing
4. The answer is "reasonable"
   - Is the list sorted?
   - Is the yellow ball yellow and round?
   - Is the area of the triangle between … and …
Classification of testing

Classification by goal:
- finding defects
- acceptance / validation
- measurement: reliability, performance, ... 

Classification by level
- system
- subsystem
- module
Acceptance test (system)

- Factory acceptance test *(FAT)*
- Site acceptance test *(SAT)*

Goals:
- is the contract fulfilled? (verification)
- is the product usable? (validation)
Measurement

- **Statistical test ("random" test)**
  "normal" input to measure how the system "normally" behaves.
  - test data selected from user profile

- **Stress test**
  How the system handles increasing / extreme load
  - graceful degrading / total collaps
  - may reveal defects

- **Profiling**
  10% of the code takes 90% of the time
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.
Black-box testing

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.

- Statement coverage
- Branch coverage
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.
tax = 0 ;
if result > 0 then
  % there is a profit
  tax = taxrate/100 * (result-threshold) ;
end if ;

result > 0

Y (there is a profit)

N

tax = taxrate/100 * (result-threshold)
if result > 0 then
  % there is a profit
  tax = taxrate/100 * (result-threshold)
else
  tax = 0
end if;

if result > 0 then
  % there is a profit
  tax = taxrate/100 * (result-threshold)
else
  tax = 0
end if;
Coverage testing flaws

Coverage testing tests code that exists, but
- not under all conditions,
- not code should exist, but doesn't.
How to do coverage testing?

- decide on test data
  - based on … (statistical, partitioning)
- use a testing tool that records
  - which code are 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.
Further coverage criteria

- Condition coverage
- Path coverage
- Data flow coverage
- Relational operator coverage
- Loop coverage

Interface level:
- Function coverage
- Call coverage
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".