

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.

Software Engineering V&V

27

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)

Software Engineering V&V

28

Black-box testing [8.1.2]

No code – but requirements!

Partition testing:

- Partitions: input and output equivalences.
 - *typical values*
 - *boundary values*
 - *invalid inputs*

Software Engineering V&V

29

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

Software Engineering V&V

30

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.

Software Engineering V&V

31

Coverage

- Statement coverage
 - every statement must be tested
- Branch coverage
 - every choice (if, while) must be tested for both true and false.

Software Engineering V&V

32

Example (coverage)

Specification

inputs: *result*, *taxrate*, *threshold*

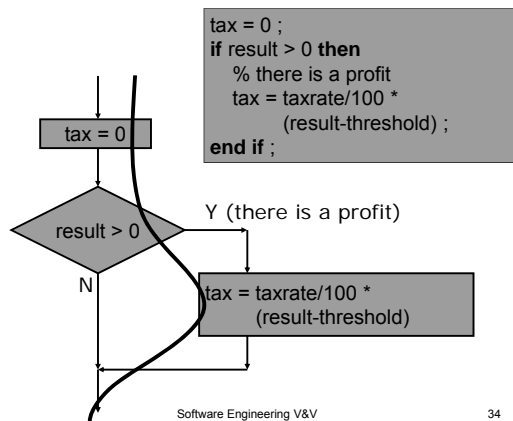
output: *tax*

relation: *tax* is $\langle \text{taxrate} \rangle$ % of the profit,
but the first $\langle \text{threshold} \rangle$ SEK is not taxed.

glossary: *profit* - a positive result.

Software Engineering V&V

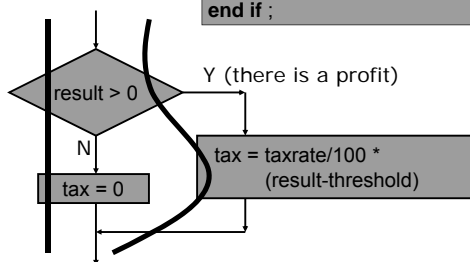
33



34

```

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

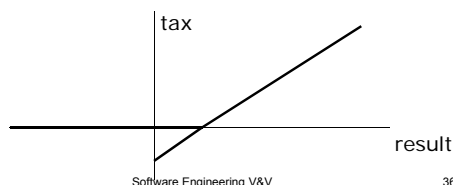


35

Coverage testing flaws

Coverage testing tests code that exists, but

- not under all conditions,
- not code that should exist, but doesn't.



36

How to do coverage testing?

- 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.

Software Engineering V&V

37

How to do coverage testing? (theoretically)

- 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*)

Software Engineering V&V

38

- Loop coverage
 - execute each loop
 - 0 times (if possible)
 - 1 time
 - several times
- Error message coverage
 - force the system to produce every error message

Software Engineering V&V

45

Interface coverage tests

- Function coverage
 - every *function* is called at least once (weak)
- Call coverage
 - every *function call* is executed at least once

Software Engineering

46

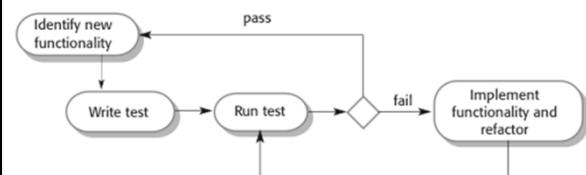
Testing concurrent systems

- 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".

Software Engineering V&V

47

Figure 8.9 Test-driven development



Summary

- Black box
 - Based on specification
 - "Natural"
- Glass box
 - Coverage ... Different criteria
 - 100% coverage \neq 100% correct
- Concurrent systems
 - Testing *may* find *some* faults

Software Engineering

49