# Structural Testing Review

Chapter 10

---

## Measuring Gaps and Redundancy

- We have seen that functional testing methods may produce test suites with serious gaps and a lot of redundancy
- Structural testing analysis allows to measure the extent of these problems

---

## Structural Metrics

- A functional testing method M produces m test cases
- A structural metric S identifies s coverage elements in the unit under test
- When the m test cases run, they traverse n coverage elements

---

## Metric definitions

- **Coverage** of method M with respect to metric S as $C(M,S) = n/s$
- **Redundancy** of method M with respect to metric S as $R(M,S) = m/s$
- **Net redundancy** of method M with respect to metric S as $NR(M,S) = m/n$

---

## Metric values for Triangle

| Method | m | n | s | C(M,S) | R(M,S) | NR(M,S) |
|---|---|---|---|---|---|---|
| Boundary Value | 15 | 7 | 11 | 0.64 | 1.36 | 2.14 |
| Worst Case Analysis | 125 | 11 | 11 | 1.00 | 11.36 | 11.36 |
| WN ECT | 4 | 4 | 11 | 0.36 | 0.36 | 1.00 |
| Decision Table | 8 | 8 | 11 | 0.72 | 0.72 | 1.00 |

---

## Metric values for Commission

| Method | m | n | s | C(M,S) | R(M,S) |
|---|---|---|---|---|---|
| Output BVA | 25 | 11 | 11 | 1 | 2.27 |
| Decision table | 3 | 11 | 11 | 1 | 0.27 |
| DD-path | 25 | 11 | 11 | 1 | 2.27 |
| DU-path | 25 | 33 | 33 | 1 | 0.76 |
| Slice | 25 | 40 | 40 | 1 | 0.63 |

## Coverage usefulness

- 100% coverage is never a guarantee of bug-free software
- Coverage reports can
  - point out inadequate test suites
  - suggest the presence of surprises, such as blind spots in the test design
  - Help identify parts of the implementation that require structural testing

7

## Coverage example

- TEX and AWK are widely used programs with comprehensive test suites
- Coverage analysis showed

| System | Segment | Branch | P-use | C-use |
|--------|---------|--------|-------|-------|
| TEX    | 85      | 72     | 53    | 48    |
| AWK    | 70      | 59     | 48    | 55    |

8

## Is 100% coverage possible?

- Short-circuit evaluation
- Mutually exclusive conditions
  - `(x > 2) || (x < 10)`
- Redundant predicates
  - `if (x == 0) do1; else do2;`
    `if (x != 0) do3; else do4;`
- Dead code
- "This should never happen"

9

## How to measure coverage?

- The source code is instrumented
- Depending on the code coverage model, code that writes to a trace file is inserted in every branch, statement etc.
- Most commercial tools measure segment and branch coverage

10

## FAQ about Coverage

- Is 100% coverage the same as exhaustive testing?
- Are branch and path coverage the same?
- Can path coverage be achieved?
- Is every path in a control flow graph testable?
- Is less than 100% coverage acceptable?
- Can I trust a test suite without measuring coverage?
- When can I stop testing?

11

## Some answers…

- When you run out of time
- When continued testing reveals no new faults
- When you cannot think of any new test cases
- When you reach a point of diminishing returns
- When mandated coverage has been attained
- When all faults have been removed

12

# A coverage counter-example

```
void Depository::give_change(int price)
{
  int n_100, n_25, n_10, n_5;
  if (deposit <= price) {
    change_due = 0;
  }
  else {
    change_due = deposit-price;
    n_100      = change_due / 100;
    change_due = change_due - n_100*100;
    n_25       = change_due / 25;
    change_due = change_due - n_25*25;
    n_10       = change_due / 10;
    change_due = change_due - n_10*10;
    n_5        = change_due / 10; // A cut-and paste bug
  }
}
```

13