Structural Testing Review

Chapter 11

The big question

- **When should testing stop?**

Possible stopping criteria

- When you run out of time
- When continued testing causes no new failures
- 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

Measuring Gaps and Redundancy

- Functional testing methods may produce test suites with serious gaps and a lot of redundancy
- Structural testing analysis makes it possible to measure the extent of these problems

– graph paths
Triangle program – nominal boundary value analysis
– worst case boundary value analysis

Paths	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
Nominal	**3**	**3**	**1**	**3**	**1**	**3**	**1**	**0**	**0**	**0**	**0**
Worst case	**5**	**12**	**6**	**11**	**6**	**12**	**7**	**17**	**18**	**19**	**12**

Structural Metrics

- What is a structural metric?
- What definitions are used for structural metrics?

Structural Metrics – 2

- 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 **c** coverage elements

Metric definitions

- **Coverage** of method M with respect to metric S is
  $C(M,S) = c / s$
  - Deals with gaps – a value < 1 means there are gaps
- **Redundancy** of method M with respect to metric S is
  $R(M,S) = m / s$
  - Deals with absolute redundancy – bigger ratio implies more redundancy – best is 1
    - Not so useful, could have massive redundancy with massive gaps giving a small ratio
- **Net redundancy** of method M with respect to metric S is
  $NR(M,S) = m / c$
  - Deals with relative redundancy – best is 1
    - Very useful, shows the redundancy of what is tested

Metric values for triangle program

Method	m	c	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 program

Method	m	c	s	C(M,S)	R(M,S)
Output BVA	25	11	11	1	2.27
Decision table	2	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 example

- TEX (Donald Knuth) and AWK (Aho, Weinberger, Kernigan) are widely used programs with comprehensive functional test suites
- Coverage analysis shows the following percentage of items covered

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

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**
- Would like to know how effective test cases are with respect to kinds of faults
  - **Can try by selecting appropriate paths**
    - **By fault type**
    - **By risk (fear)**

Is 100% coverage possible?

- Can you suggest cases that prevent 100% coverage?

Is 100% coverage possible? – 2

- Lazy (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"
    ```

How to measure coverage?

- **Can you suggest ways to measure coverage?**

How to measure coverage? – 2

- 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

Questions 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?

Coverage counter-example vending machine

```
void give_change(int price, deposit) {
  int n_100, n_25, n_10, n_5, change_due;
  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; // Cut-and-paste bug
  }
}
```

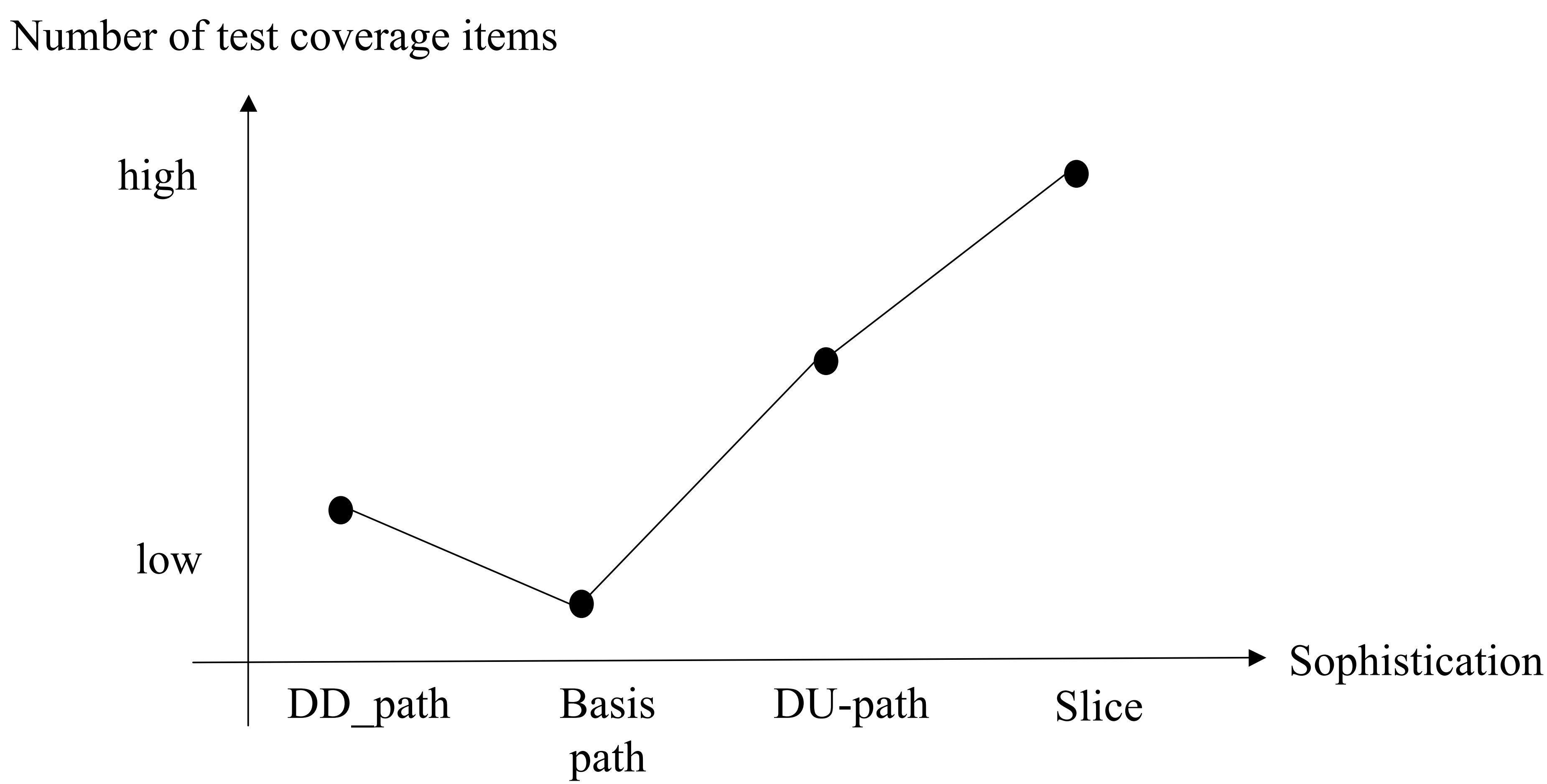
Cannot guarantee path will use revealing test values for deposit and price

Coverage counter-example aircraft control

```
void flight_control_event_handler (event e) {
  switch(e)
  { ...
    case RAISE_LANDING_GEAR:
      landing_gear_motor ( turn_on_until_raised );
      break;
    ...
  }
}
```

Can you find the bug?
Will any path test find the bug?
What can correct the bug?

Trend line test coverage of items



Trend line test method effort