Introduction to Software Testing
Chapter 6
Input Space Partition Testing

Paul Ammann & Jeff Offutt

http://www.cs.gmu.edu/~offutt/softwaretest/

Modeling the Input Domain

- Step 1 : Identify testable functions
  - Individual methods have one testable function
  - Methods in a class often have the same characteristics
  - Programs have more complicated characteristics—modeling documents such as UML can be used to design characteristics
  - Systems of integrated hardware and software components can use devices, operating systems, hardware platforms, browsers, etc.

```
public enum Triangle { Scalene, Isosceles, Equilateral, Invalid }

public static Triangle triang (int Side, int Side2, int Side3)

// Side1, Side2, and Side3 represent the lengths of the sides of a triangle

// Returns the appropriate enum value
```

Modeling the Input Domain

- Step 2 : Find all the parameters
  - Often fairly straightforward, even mechanical
  - Important to be complete
  - Methods : Parameters and **state** (non-local) **variables** used
  - Components : Parameters to methods and state variables
  - System : All inputs, including files and databases
- State variables?
  - TreeSet.add(E e)

Modeling the Input Domain (*cont*)

- Step 3 : Model the input domain
  - The domain is scoped by the parameters
  - The structure is defined in terms of characteristics
  - Each characteristic is partitioned into sets of blocks
  - Each block represents a set of values
  - This is the most creative design step in using ISP
- Step 4 : Apply a test criterion to choose combinations of values
  - A test input has a value for each parameter
  - One block for each characteristic
  - Choosing all combinations is usually infeasible
  - Coverage criteria allow subsets to be chosen
- Step 5 : Refine combinations of blocks into test inputs
  - Choose appropriate values from each block

Step 4 – Choosing Combinations of Values (6.2)

- Once characteristics and partitions are defined, the next step is to choose test values
- We use criteria – to choose effective subsets
- The most obvious criterion is to choose all combinations

All Combinations (ACoC) : All combinations of blocks from all characteristics must be used.

- Number of tests is the product of the number of blocks in each characteristic : $\prod_{i=1}^{Q}(B_i)$
- The second characterization of triang() results in 4*4*4 = 64 tests
  - Too many ?

ISP Criteria – All Combinations

- Consider the "second characterization" of Triang as given before:

Characteristic	b_1	b_2	b_3	b_4
q_1 = "Refinement of q_1"	greater than 1	equal to 1	equal to 0	less than 0
q_2 = "Refinement of q_2"	greater than 1	equal to 1	equal to 0	less than 0
q_3 = "Refinement of q_3"	greater than 1	equal to 1	equal to 0	less than 0

- For convenience, we relabel the blocks using abstractions:

Characteristic	b_1	b_2	b_3	b_4
A	A1	A2	A3	A4
B	B1	B2	B3	B4
C	C1	C2	C3	C4

ISP Criteria – ACoC Tests

A1 B1 C1	A2 B1 C1	A3 B1 C1	A4 B1 C1
A1 B1 C2	A2 B1 C2	A3 B1 C2	A4 B1 C2
A1 B1 C3	A2 B1 C3	A3 B1 C3	A4 B1 C3
A1 B1 C4	A2 B1 C4	A3 B1 C4	A4 B1 C4
A1 B2 C1	A2 B2 C1	A3 B2 C1	A4 B2 C1
A1 B2 C2	A2 B2 C2	A3 B2 C2	A4 B2 C2
A1 B2 C3	A2 B2 C3	A3 B2 C3	A4 B2 C3
A1 B2 C4	A2 B2 C4	A3 B2 C4	A4 B2 C4
A1 B3 C1	A2 B3 C1	A3 B3 C1	A4 B3 C1
A1 B3 C2	A2 B3 C2	A3 B3 C2	A4 B3 C2
A1 B3 C3	A2 B3 C3	A3 B3 C3	A4 B3 C3
A1 B3 C4	A2 B3 C4	A3 B3 C4	A4 B3 C4
A1 B4 C1	A2 B4 C1	A3 B4 C1	A4 B4 C1
A1 B4 C2	A2 B4 C2	A3 B4 C2	A4 B4 C2
A1 B4 C3	A2 B4 C3	A3 B4 C3	A4 B4 C3
A1 B4 C4	A2 B4 C4	A3 B4 C4	A4 B4 C4

ACoC yields 4*4*4 = 64 tests for Triang!

This is almost certainly more than we need

Only 8 are valid (all sides greater than zero)

Pairwise and N-way Testing

- **Exercise:** Consider that we have been assigned to test a web application. That web application:

Clients	Browsers	Languages	Databases	Servers
Windows A	Browser A	English	Database A	Unix
Windows B	Browser B	French	Database B	Linux
Windows C	Browser C	German	Database C	Solaris
Mac A	Browser D	Arabic		HPUX
Mac B		Mandarin		
Mac C				

Pairwise and N-way Testing

- N-way testing relies on the assumption that many faults are caused by interactions between a relatively small number of parameters.
- A study has been done to analyze real reported defects within 4 different types of applications, on a time span of 15 years.
- The studied applications were medical devices, a NASA project, a web server, a browser.
- For medical applications: Only one defect was triggered by interactions of 4 parameter values.
- For the web server: Almost 90% of the defects were caused by three or fewer parameter interactions.

2-way Interaction Failure

```
if (pressure < 10) {
        // do something
        if (volume > 300)   {
                faulty code!   BOOM!
        }
        else {
                good code, no problem
        }
}
else {
        // do something else
}
```

Combinatorial Testing – Input Values

- Let's consider 3 input variables X, Y, and Z.
- X has two possible values: x1, x2
- Y has two possible values: y1, y2
- Z has two possible values: z1, z2
- All possible combinations:
  - x1 y1 z1
  - x1 y1 z2
  - x1 y2 z1
  - x1 y2 z2
  - x2 y1 z1
  - x2 y1 z2
  - x2 y2 z1
  - x2 y2 z2

What if we want to make sure that each pair of values appears at least in one test case?

Instead of making all possible combinations appear in our tests.

Combinatorial Testing – Input Values

- All combinations:
  - x1 y1 z1
  - x1 y1 z2
  - x1 y2 z1
  - x1 y2 z2
  - x2 y1 z1
  - x2 y1 z2
  - x2 y2 z1
  - x2 y2 z2

What pairs do we have?

x1 y1
x1 y2
x1 z1
x1 z2
x2 y1
x2 y2
x2 z1
x2 z2
y1 z1
y1 z2
y2 z1
y2 z2

Combinatorial Testing – Input Values

- All combinations:
  - x1 y1 z1
  - x1 y1 z2
  - x1 y2 z1
  - x1 y2 z2
  - x2 y1 z1
  - x2 y1 z2
  - x2 y2 z1
  - x2 y2 z2

What pairs do we have?

x1 y1
x1 y2
x1 z1
x1 z2
x2 y1
x2 y2
x2 z1
x2 z2
y1 z1
y1 z2
y2 z1
y2 z2

Combinatorial Testing – Input Values

- All combinations:
  - x1 y1 z1
  - x1 y1 z2
  - x1 y2 z1
  - x1 y2 z2
  - x2 y1 z1
  - x2 y1 z2
  - x2 y2 z1
  - x2 y2 z2

What pairs do we have?

x1 y1
x1 y2
x1 z1
x1 z2
x2 y1
x2 y2
x2 z1
x2 z2
y1 z1
y1 z2
y2 z1
y2 z2

Combinatorial Testing – Input Values

- All combinations:
  - x1 y1 z1
  - x1 y1 z2
  - x1 y2 z1
  - x1 y2 z2
  - x2 y1 z1
  - x2 y1 z2
  - x2 y2 z1
  - x2 y2 z2

What pairs do we have?

x1 y1
x1 y2
x1 z1
x1 z2
x2 y1
x2 y2
x2 z1
x2 z2
y1 z1
y1 z2
y2 z1
y2 z2

Combinatorial Testing – Input Values

- All combinations:
  - x1 y1 z1
  - x1 y1 z2
  - x1 y2 z1
  - x1 y2 z2
  - x2 y1 z1
  - x2 y1 z2
  - x2 y2 z1
  - x2 y2 z2

What pairs do we have?

x1 y1
x1 y2
x1 z1
x1 z2
x2 y1
x2 y2
x2 z1
x2 z2
y1 z1
y1 z2
y2 z1
y2 z2

ISP Criteria – Pair-Wise

- Each choice yields few tests—cheap but maybe ineffective
- Another approach combines values with other values

Pair-Wise Coverage (PWC) : A value from each block for each characteristic must be combined with a value from every block for each other characteristic.

- Number of **test values** is at least the product of two largest characteristics

$$(\text{Max}^{Q}_{i=1}(B_i)) * (\text{Max}^{Q}_{j=1,\, j!=i}(B_j))$$

ISP Criteria – Pair-Wise

- Each choice yields few tests—cheap but maybe ineffective
- Another approach combines values with other values

Pair-Wise Coverage (PWC) : A value from each block for each characteristic must be combined with a value from every block for each other characteristic.

- Number of **test values** is at least the product of two largest characteristics

$$(\text{Max}^{Q}_{i=1}(B_i)) * (\text{Max}^{Q}_{j=1,\, j!=i}(B_j))$$

For *triang*() :

A1, B1, C1	A1, B2, C2	A1, B3, C3	A1, B4, C4
A2, B1, C2	A2, B2, C3	A2, B3, C4	A2, B4, C1
A3, B1, C3	A3, B2, C4	A3, B3, C1	A3, B4, C2
A4, B1, C4	A4, B2, C1	A4, B3, C2	A4, B4, C3

Write down PWC tests
Use the abstract labels
(Hint: Should be 16 tests)

ISP Criteria –T-Wise

- A natural extension is to require combinations of *t* values instead of *2*

t-Wise Coverage (TWC) : A value from each block for each group of t characteristics must be combined.

- Number of tests is at least the product of *t* largest characteristics
- If all characteristics are the same size, the formula is

$$(\text{Max}_{i=1}^{Q}(B_i))^t$$

- If *t* is the number of characteristics *Q*, then all combinations
- That is … *Q-wise = AC*
- *t*-wise is expensive and benefits are not clear

Issues with pair-wise and t-wise?

Required Reading

- Chapter 6 from Amman's and Offut's book: An Introduction to Software Testing