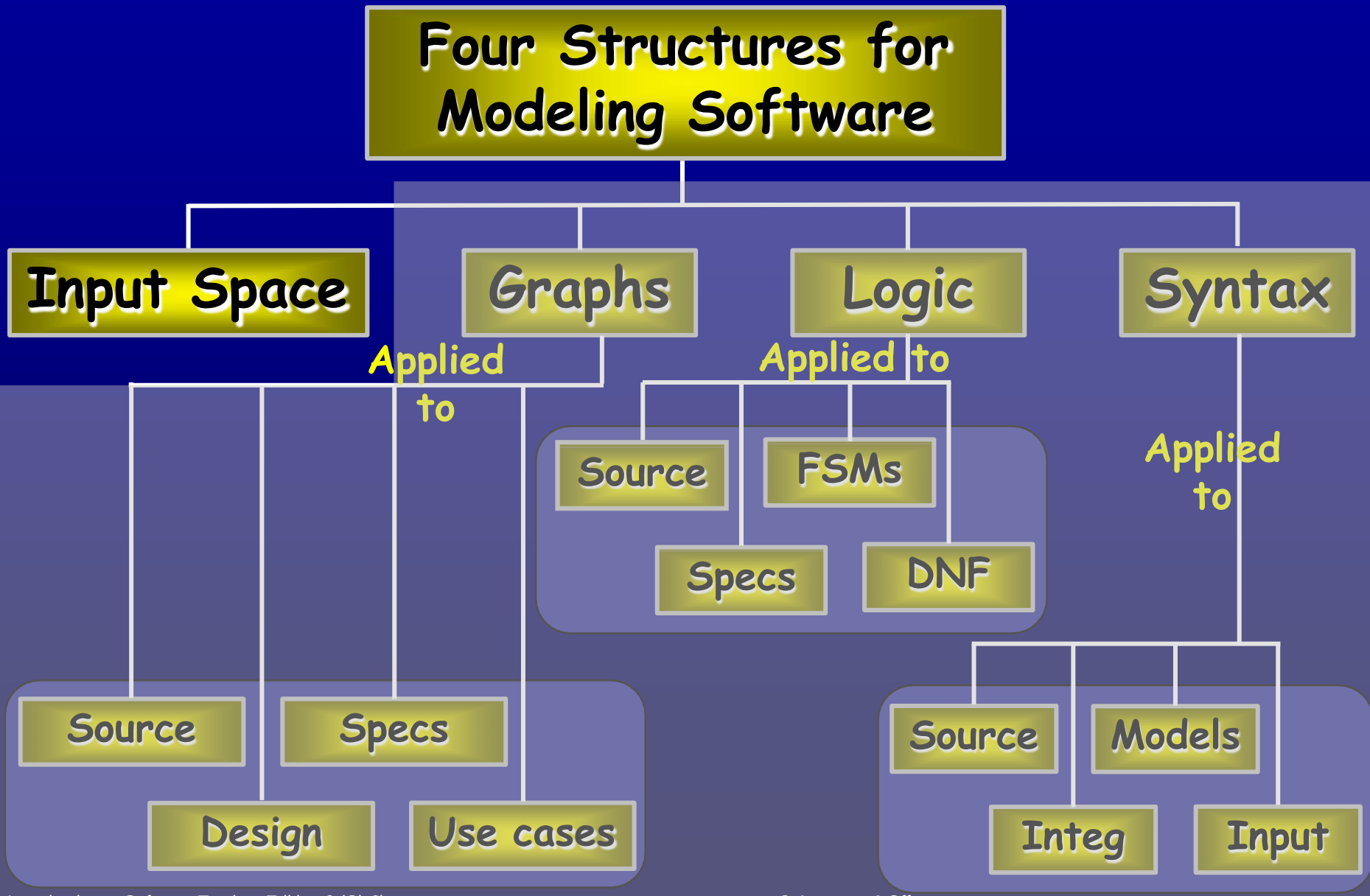
Introduction to Software Testing
Chapter 6
Input Space Partition Testing

Paul Ammann & Jeff Offutt

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

Ch. 6 : Input Space Coverage



ISP Criteria – Each Choice

- 64 tests for triang() is almost certainly way too many
- One criterion comes from the idea that we should try at least one value from each block

Each Choice Coverage (ECC) : One value from each block for each characteristic must be used in at least one test case.

- Number of tests is the number of blocks in the largest characteristic : $\mathbf{Max}_{i=1}^{Q}(B_i)$

For *triang*() : A1, B1, C1
A2, B2, C2
A3, B3, C3
A4, B4, C4

Write down EC tests
Use the abstract labels (A1, A2, ...)

Substituting values: 2, 2, 2
1, 1, 1
0, 0, 0
-1, -1, -1

Suggest values ...
Limitations?

Example

- Given a system with three characteristics.
- The first characteristic has the blocks [A, B]
- The second characteristic has the blocks [1, 2, 3],
- The third characteristic has the blocks [x, y]
- ECC can be satisfied in many ways, including the three tests (A, 1, x), (B, 2, y), and (--, 3, x)

ISP Criteria – Base Choice

- Testers sometimes recognize that certain values are important
- This uses domain knowledge of the program

Base Choice Coverage (BCC) : A base choice block is chosen for each characteristic, and a base test is formed by using the base choice for each characteristic. Subsequent tests are chosen by holding all but one base choice constant and using each non-base choice in each other characteristic.

- Number of tests is one base test + one test for each other block $1 + \sum_{i=1}^{Q} (B_i - 1)$

ISP Criteria – Base Choice

- Testers sometimes recognize that certain values are important
- This uses domain knowledge of the program

Base Choice Coverage (BCC) : **A base choice block is chosen for each characteristic, and a base test is formed by using the base choice for each characteristic. Subsequent tests are chosen by holding all but one base choice constant and using each non-base choice in each other characteristic.**

- Number of tests is one base test + one test for each other block $1 + \sum_{i=1}^{Q} (B_i - 1)$

For *triang*() :

Base **A1, B1, C1**

Write down BCC tests

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

Example

- Given a system with three characteristics.
- The first characteristic has the blocks [A, B]
- The second characteristic has the blocks [1, 2, 3],
- The third characteristic has the blocks [x, y]
- Suppose that the base choice blocks are 'A', '1', 'x'
- The base choice test is [A, 1,x], and the following additional tests would be needed:
- (B, 1, x)
  (A, 2, x)
  (A, 3, x)
  (A, 1, y)

Base Choice Notes

- The base test must be feasible
  - That is, all base choices must be compatible
- Base choices can be
  - Most likely from an end-use point of view
  - Simplest
  - Smallest
  - First in some ordering
- Happy path tests often make good base choices
- The base choice is a crucial design decision
  - Test designers should document why the choices were made

ISP Criteria – Multiple Base Choice

- We sometimes have more than one logical base choice

Multiple Base Choice Coverage (MBCC) :At least one, and possibly more, base choice blocks are chosen for each characteristic, and base tests are formed by using each base choice for each characteristic at least once. Subsequent tests are chosen by holding all but one base choice constant for each base test and using each non-base choice in each other characteristic.

- If M base tests and m_i base choices for each characteristic:

$$M + \sum_{i=1}^{Q} (M * (B_i - m_i))$$

For *triang*() :Bases

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

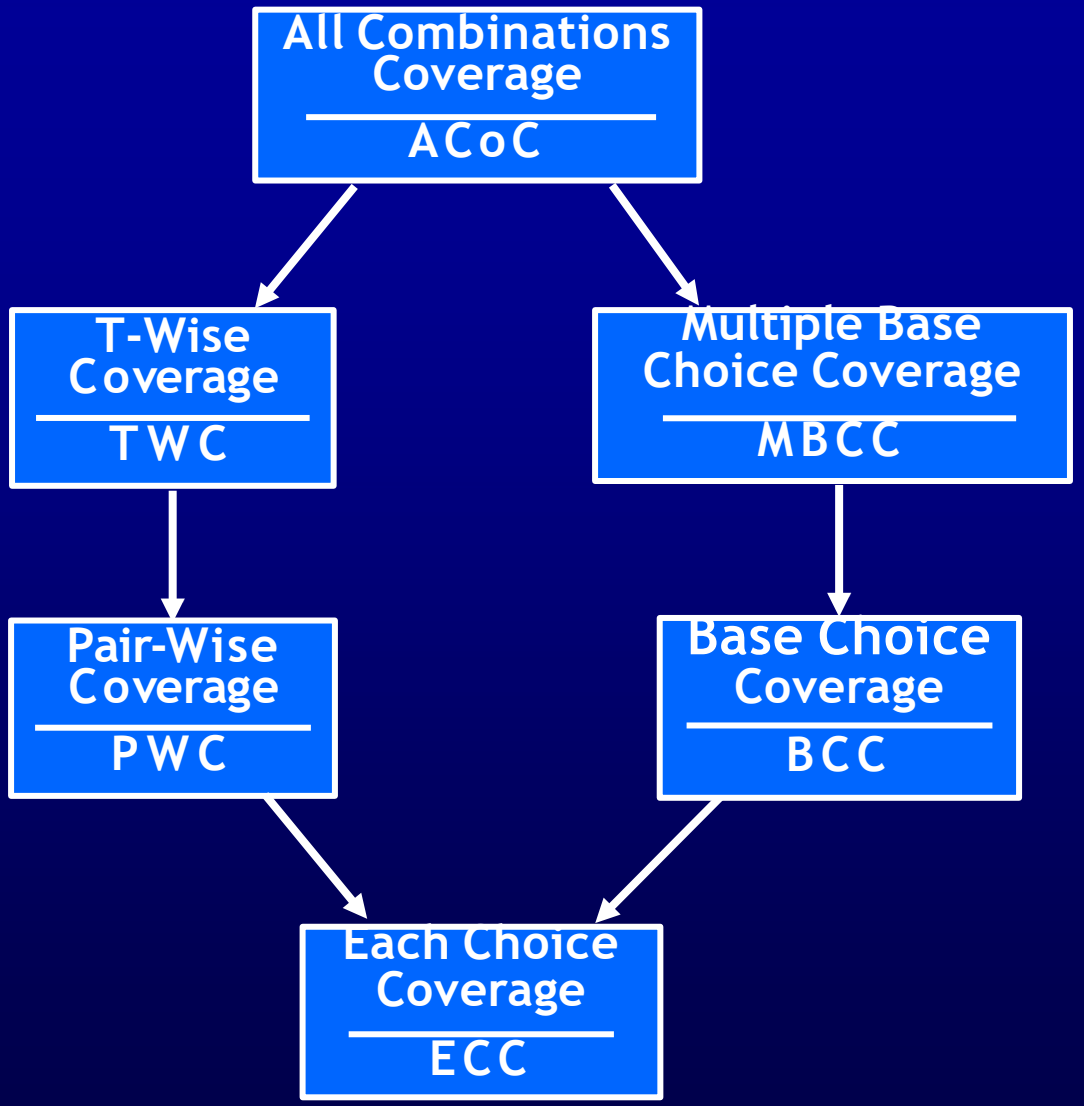
Example

- Given our triang() example with three sides
- Assume that we decide to include two base choices for side 1 in triang (), "greater than 1" and "equal to 1."
- This would result in the two base tests (2, 2, 2) and (1, 2, 2).
  - **If *M* base tests and m_i base choices for each characteristic:**

$$M + \sum_{i=1}^{Q} (M * (B_i - m_i))$$

- The formula above is thus evaluated with $M = 2$, $m1 = 2$, and $mi = 1 \; \forall \; i, \; 1 < i \leq 3$. That is, 2 + (2*(4-2)) + (2*(4-1)) + (2*(4-1)) = 18.

ISP Coverage Criteria Subsumption



Constraints Among Characteristics (6.3)

- Some combinations of blocks are infeasible
  - "less than zero" and "scalene" ... not possible at the same time
- These are represented as constraints among blocks
- Two general types of constraints
  - A block from one characteristic cannot be combined with a specific block from another
  - A block from one characteristic can ONLY BE combined with a specific block form another characteristic
- Handling constraints depends on the criterion used
  - ACC, PWC, TWC : Drop the infeasible pairs
  - BCC, MBCC : Change a value to another non-base choice to find a feasible combination

Example Handling Constraints

```
public boolean findElement (List list, Object element)
// Effects: if list or element is null throw NullPointerException
//          else return true if element is in the list, false otherwise
```

Characteristic	Block 1	Block 2	Block 3	Block 4
A : length and contents	One element	More than one, unsorted	More than one, sorted	More than one, all identical
B : match	element not found	element found once	element found more than once	
Invalid combinations : (A1, B3), (A4, B2)				

element cannot be in a one-element list more than once

If the list only has one element, but it appears multiple times, we cannot find it just once

Input Space Partitioning Summary

- Fairly easy to apply, even with no automation
- Convenient ways to add more or less testing
- Applicable to all levels of testing - unit, class, integration, system, etc.
- Based only on the input space of the program, not the implementation

Simple, straightforward, effective, and widely used