Course Notes Set 12-a:

Combinatorial Interaction Testing

Computer Science and Software Engineering
Auburn University

Combinatorial Interaction Testing

- Execution requires interactions with external entities, e.g., users, sensors, or other systems
- System behaviors depend on the inputs from external entities
- Example:
  - A system may allow user configuration options, e.g., 160 binary options, 10 ternary, 5 4-setting, 5 6 choices ➔ $2^{160} \times 3^{10} \times 4^{5} \times 6^{5} \sim \infty$
  - Impossible to test all configurations
  - Really?
  - Filing tax return through Form 1040

Combinatorial Interaction Testing

- Impossible to test all configurations or possibilities
- The *sampling* of the configurations is called *combinatorial interaction testing (CIT)*

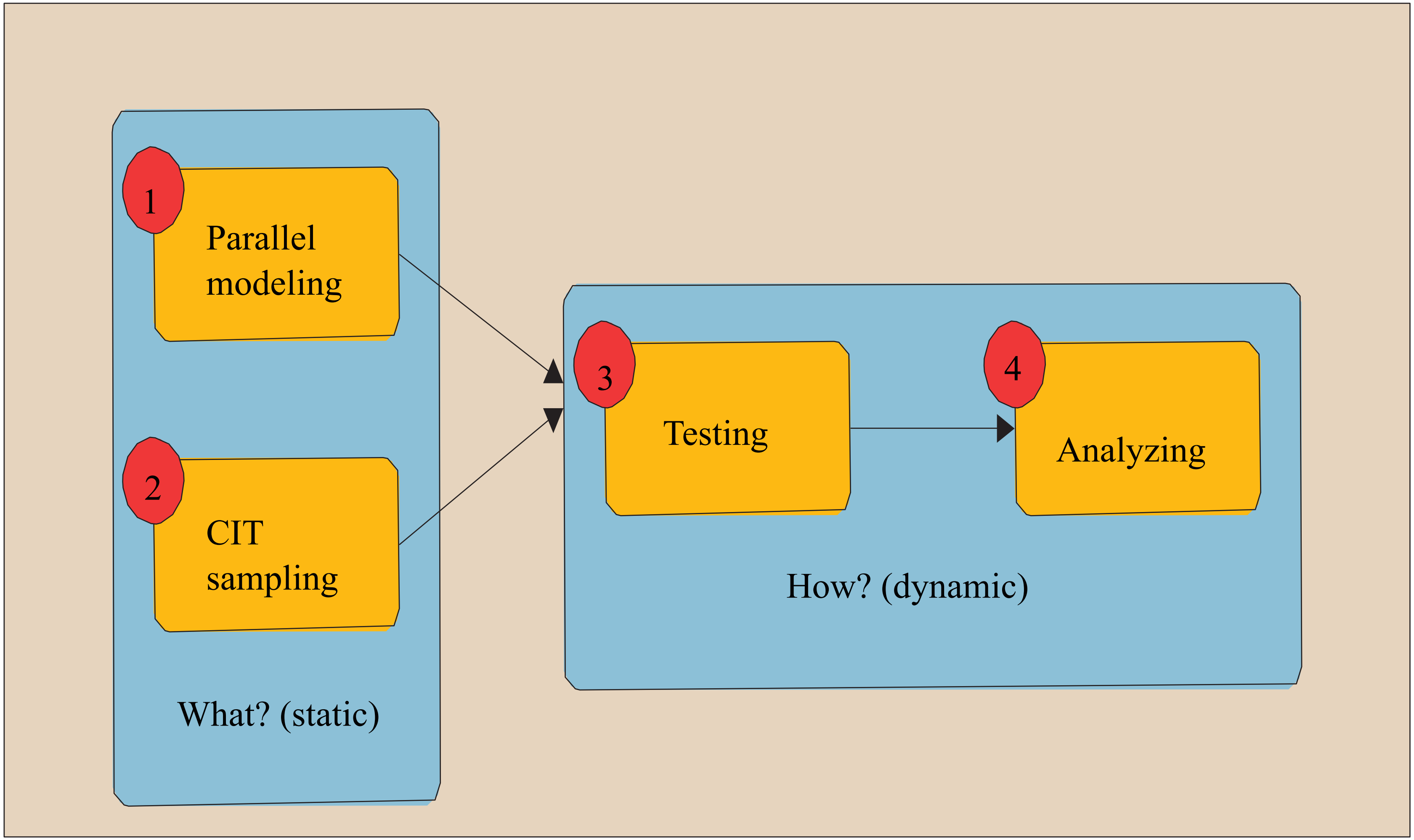
- A system under test has to be modeled based on the interactions into *factors*

- Goal: to generate a set of combinations of factors and their values that satisfy certain requirements

Combinatorial Interaction Testing

- Requirement example
  - *Pairwise testing* – all possible combinations between two factors appear at least once in the sample
  - A simplified tax return example:
  - **S**tatus: Single, married-jointly, married-separately
  - **D**isabled: Yes, No
  - **T**ax ranges: 0 – 20K, 20K-40K, 40K-∞
  - Pairwise: S&D → 6
    S&T → 9

Combinatorial Interaction Testing



Combinatorial Interaction Testing

- Modeling
  - input space is represented in a set of factors
  - Best to have a limited set of values for each factor
    - Continuous input factors can be discretized, e.g., income value ➔ 0 – 20K, 20K-40K, 40K-∞
    - Still too many values, use equivalence partitioning
  - There may be constraints or dependency among factors
    - Factor-1: TCP/IP = *True* ➔ Factor-2: Network-Enabled must be *True*
    - Factor-2: Network-Enabled = *False* ➔ Factor-5: Remote-Printing cannot be *True*

Combinatorial Interaction Testing

- Seed models
  - Must-include-seeds: certain combinations
  - Must-avoid-seeds: already tested combinations

- **Sampling**
  - Computing an efficient combinatorial objects, called covering array, to satisfy a given coverage criterion

Combinatorial Interaction Testing

- Covering Array
  - t-way covering array: for each set of t factors, every possible combination appears at least once
  - A, B, C: 0, 1 D, E: 0, 1, 2
  - Total configurations: 2 X 2 X 2 X 3 X 3 = 72
  - 2-way covering array

A	B	C	D	E
0	1	1	2	0
0	0	0	0	0
0	0	0	1	1
1	1	1	0	1
0	1	0	0	2
1	0	1	1	0
1	1	1	1	2
1	0	0	2	1
1	0	0	2	2

2-way covering array

- ➢Only 9 configurations are needed for 2-way covering array
- ➢t is called the *coverage strength*, 2 in this case
- ➢Study has shown that low strength coverage correlate to high statement and branch coverage
- ➢In practice, the needed t is much smaller than the number of factors. Typically, 2=< t =< 6, with t = 2 being the most typical case

A	B		B	C
0	0		0	0
0	1		0	1
1	0		1	0
1	1		1	1

A	B	C
0	0	0
0	1	0
1	0	1
1	1	1
0	*	1
1	*	0

- Process one pair at a time
- Only 6 configurations are needed

Combinatorial Interaction Testing

- **Variable-strength covering arrays**
  - A subset of factors must be tested more thoroughly
  - Most factors can be tested with t-strength covering array (fixed strength), while a subset set can be tested with >t-strength (variable strength)

Combinatorial Interaction Testing

- **Test case-aware covering arrays**:
  - Covering arrays are designed for each system configuration
  - Each configuration can be tested by multiple test cases and one test case may run under different configurations
  - Some test cases can only be run when certain conditions are true or false – constraints
  - E.g., testing remote printing … network must be enabled
  - If not considered carefully, a set of test cases may not be executed

Combinatorial Interaction Testing

- Some issues to consider
  - Cost-aware covering arrays
    - Some configurations are more expensive to run, e.g., needs additional installation and compilation
    - Needs to “order” test configurations
  - Incremental Covering arrays
    - Start form pre-defined seeds
    - Start from (t-1)-way array as seed. Use the existing configurations and add only a small amount of new configurations to achieve t-way array
  - Prioritization based on cost, execution frequency, etc.

This slide set is based on *Moving Forward with Combinatorial Interaction Testing*, by C. Yilmaz, S. Fouche, M. Cohen, A. Porter, G. Demiroz, and U. Koc, in Computer, February, 2014, Vol. 47, No.2, pp. 37 - 45