








# CUI Abbottabad

Department of Computer Science

## SOFTWARE TESTING

### **Lecture 16**

### **Advanced Black box Testing Combinatorial testing and Pairwise Testing**

- 
- ▶ **Combinatorial testing:** A means to identify a suitable subset of test combinations to achieve a predetermined level of coverage when testing an object with multiple parameters and where those parameters themselves each have several values, which gives rise to more combinations than are feasible to test in the time allowed.
  - ▶ Combinatorial explosion occurs when a test problem can be described as a set of parameters with a number of different values and the total number of possible combinations of parameter values is too large to be feasibly tested. The main purpose of combinatorial testing is to identify a manageable set of (interesting) combinations.
  - ▶ The two most popular combinatorial test techniques [International software testing qualification board (ISTQB)] are discussed: Classification Tree Method and Pairwise Testing.
- 
- 
- 
- 

# PAIRWISE TESTING

- ▶ Pairwise testing: A black box test design technique in which test cases are designed to execute all possible discrete combinations of each pair of input parameters.
- ▶ Pairwise testing is another technique that can be used for testing combinations of values.
- ▶ Pairwise testing is a technique that identifies all pairs of values from the total input domain.
- ▶ By using this technique, the number of tests can be reduced while still having confidence in the coverage.

# EXAMPLE

Suppose we want to demonstrate that a new software application works correctly on PCs that use the either Windows or Linux as an operating systems, either Intel or AMD processors, and the IPv4 or IPv6 protocols. This is a total of  $2 \times 2 \times 2 = 8$  possibilities but, as the table below shows, only four tests are required to test **every component interacting with every other component at least once** (check!).

Test case	Operating System	Processor	Protocol
1	Windows	Intel	IPv4
2	Windows	AMD	IPv6
3	Linux	Intel	IPv6
4	Linux	AMD	IPv4

# ORTHOGONAL ARRAY TESTING

- ▶ A common method of deriving pairwise tests is by using orthogonal arrays.
- ▶ Orthogonal Array Testing is a black box testing method wherein, the test data is large and consist of permutations ad combinations.

$$L_{\text{Runs}}(\text{Levels}^{\text{Factors}})$$

- ▶ Runs(N)- Number of rows in the array, which translates into a number of test cases generated.
- ▶ Factors(K)- Number of columns in the array, which translates into max number of variables.
- ▶ Levels (V) – Max number of the values that can be taken by a single variable or factor.

- ▶ Step 1: Identify the maximum number of independent input variables with which a system will be tested. This will map to the factors of the array—each input variable maps to a different factor.
- ▶ Step 2: Identify the maximum number of values that each independent variable will take. This will map to the levels of the array.
- ▶ Step 3: Find a suitable orthogonal array with the smallest number of runs. The number of runs can be derived from various websites.
- ▶ Step 4: Map the variables to the factors and values of each variable to the levels on the array.
- ▶ Step 5: Check for any “left-over” levels in the array that have not been mapped. Choose arbitrary valid values for those left-over levels. You can perform this step in a round robin fashion.
- ▶ Step 6: Translate them into suitable test cases.

# EXAMPLE

- ▶ Suppose you are testing a form which have following fields: age, qualification, language, and location. Age can be less than 18, more than 18 and more than 60. Qualification contains High school, graduate, post graduate. Language can be English, Urdu, and Persian. Location includes atd, isb, and khi.
- ▶ **Identify the Factors**
  - ▶ Age
  - ▶ Qualification
  - ▶ Language
  - ▶ Location
- ▶ **Identify the levels**
  - ▶ We see each factors contains 3 values/Levels

# EXAMPLE

- ▶ Visit [http://support.sas.com/techsup/technote/ts723\\_Designs.txt](http://support.sas.com/techsup/technote/ts723_Designs.txt) to see the runs for your scenario.
- ▶ According to Lruns formula  $\text{Levels}^{\text{factor}}$  so in our scenario its  $3^4$ .
- ▶ We have 9 runs so 9 test cases will be generated instead of 81 for full coverage testing.
- ▶ Draw an array with 9 runs and 4 factors.
- ▶ Fill the table with values from [http://support.sas.com/techsup/technote/ts723\\_Designs.txt](http://support.sas.com/techsup/technote/ts723_Designs.txt)
- ▶  $L_9(3^4)$



Runs	Factor 1	Factor 2	Factor 3	Factor 4
1	0	0	0	0
2	0	1	2	1
3	0	2	1	2
4	1	0	2	2
5	1	1	1	0
6	1	2	0	1
7	2	0	1	1
8	2	1	0	2
9	2	2	2	0

# EXAMPLE

- ▶ Fill the table with actual data
- ▶ Factors and assign their values 0, 1 and 2
- ▶ Factor 1 Age:
  - ▶ Less than 18=0
  - ▶ More than 18=1
  - ▶ More than 60=2
- ▶ Factor 2 = qualification
  - ▶ High school = 0
  - ▶ Graduate = 1
  - ▶ Post Graduate =2

# EXAMPLE

- ▶ Factor 3= language
  - ▶ English = 0
  - ▶ Urdu = 1
  - ▶ Persian = 2
- ▶ Factor 4 = Location
  - ▶ Atd = 0
  - ▶ Isb = 1
  - ▶ Khi = 2

Runs	Age	Qualification	Language	Location
TC 1	Less than 18	High school	English	Atd
TC 2	Less than 18	Graduate	Persian	Isb
TC 3	Less than 18	Post Graduate	Urdu	Khi
TC 4	More than 18	High school	Persian	Khi
TC 5	More than 18	Graduate	Urdu	Atd
TC 6	More than 18	Post Graduate	English	Isb
TC 7	More than 60	High school	Urdu	Isb
TC 8	More than 60	Graduate	English	Khi
TC 9	More than 60	Post Graduate	Persian	Atd