1

CS 5387
SOFTWARE V&V

LECTURE 7
BLACK-BOX TESTING

1

Outline

2

- Quiz
- Black-Box Testing
  - Equivalence Class Testing (Equivalence Partitioning)
  - Boundary value analysis
  - Decision Table Testing

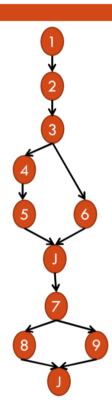
2

Quiz - 1

3

1. **read** (x)
2. **read** (z)
3. **if** x != 0 **then** {
4. y = x * z;
5. x = z }
6. **else** y = z
7. **if** y > 1 **then**
8. **print** (y * z)
9. **else** **print** (y * x)

Cyclomatic complexity:
- *Upper limit on the number of test cases required to ensure branch coverage*
- Cyclomatic complexity= E – N + 2 = 12 – 11 + 2 = 3
- B + 1 = 2 + 1 = 3



3

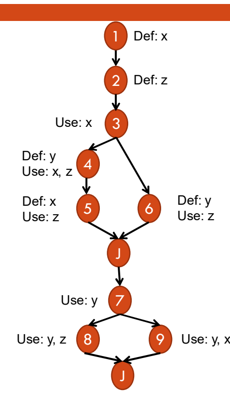
Quiz – 2

4

- **Def-use coverage**: every path from every definition of every variable to every use of that definition is exercised in some test.
  1. **read** (x)
  2. **read** (z)
  3. **if** x != 0 **then** {
  4. y = x * z;
  5. x = z }
  6. **else** y = z
  7. **if** y > 1 **then**
  8. **print** (y * z)
  9. **else** **print** (y * x)

Minimal set with respect to X:
{X, Z} = {0, 1} (1, 2, 3, 6, J, 7, 9, J)
{X, Z} = {1, 1} (1, 2, 3, 4, 5, J, 7, 9, J)

Minimal set with respect to Y:
{X, Z} = {1, 2} (1, 2, 3, 4, 5, J, 7, 8, J)
{X, Z} = {1, 1} (1, 2, 3, 4, 5, J, 7, 9, J)
{X, Z} = {0, 2} (1, 2, 3, 6, J, 7, 8)
{X, Z} = {0, 1} (1, 2, 3, 6, J, 7, 9, J)



4

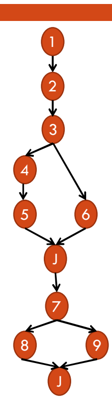
Quiz - 3

5

- **Path Coverage: Every distinct path through code is executed at least once**

```
1. read (x)
2. read (z)
3. if x != 0 then {
4.     y = x * z;
5.     x = z }
6. else  y = z
7. if y > 1 then
8.     print (y * z)
9. else   print (y * x)
```

Test Paths:
1, 2, 3, 4, 5, J, 7, 8, J. Test Case (x, z): {1, 2}
1, 2, 3, 4, 5, J, 7, 9, J. Test Case (x, z): {1, 1}
1, 2, 3, 6, J, 7, 8, J. Test Case (x, z): {0, 2}
1, 2, 3, 6, J, 7, 9, J. Test Case (x, z): {0, 1}

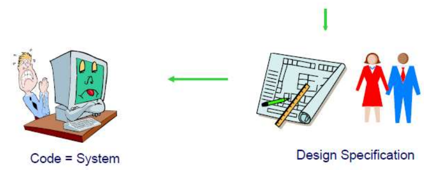


5

Unit & Integration Testing

6

- Objective: to ensure that code implemented the design properly.



6

Unit Testing

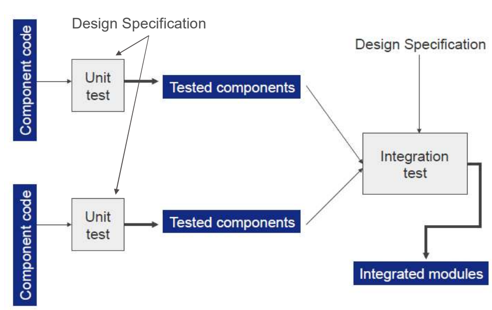
7

- Code Inspections
- Code Walkthroughs
- Black-box Testing
- White-box Testing

7

Unit and Integration Testing

8

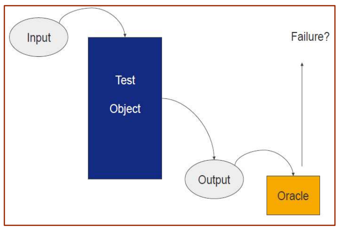


8

Test Oracles

9

- Two types of oracles
  - Human: an expert that can examine an input and its associated output and determine whether the program delivered the correct output for this particular input.
  - Automated: a system capable of performing the above task.

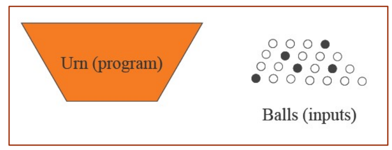


9

Balls and Urn - 1

10

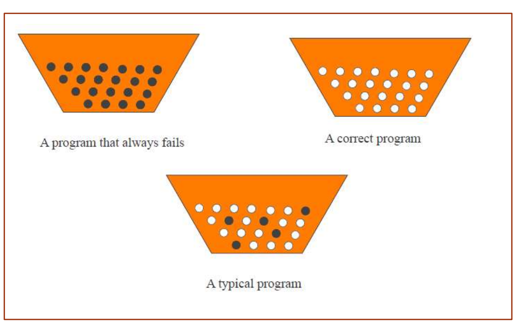
- Testing can be viewed as selecting different colored balls from an urn where:
  - Black ball = input on which program fails.
  - White ball = input on which program succeeds.
  - Only when testing is exhaustive is there an “empty” urn.



10

Balls and Urn - 2

11

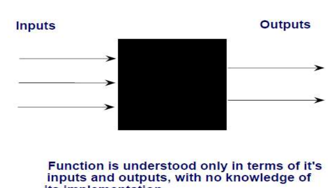


11

Black-box/Closed-box Testing

12

- Incorrect or missing functions
- Interface errors
- Performance error



12

Black-Box Testing Techniques

13

- Definition: a strategy in which testing is based on requirements and specifications.
- Applicability: all levels of system development
  - Unit
  - Integration
  - System
  - Acceptance
- Disadvantages: never be sure of how much of the system under test (SUT) has been tested.
- Advantages: directs tester to choose subsets to tests that are both efficient and effective in finding defects *early in development*.

13

Black-Box Testing

14

- Exhaustive testing
- Equivalence class testing
- Boundary value analysis
- Decision table testing
- Pairwise testing
- State-Transition testing
- Domain analysis testing
- Use case testing

For Today

14

Exhaustive Testing

15

- Definition: testing with every member of the input value space.
- Input value space: the set of all possible input values to the program.
- How Feasible?

```
int increment (int a) {
............
}
```

If one test takes 1 ms how much time it will take to test exhaustively function above?

No of possible inputs: From −2,147,483,648 to 2,147,483,647, from $-(2^{31})$ to $2^{31} - 1 = 2^{32} - 1$ possible inputs

Time to test $2^{32} - 1$ ms = $(2^{32} - 1)/1000$ sec = 1.6 months

```
long increment (long a) {
............
}
```

Time to test $2^{64} - 1$ ms = $(2^{64} - 1)/1000$ sec = 584 542 046,1 years

15

Equivalence Class Testing

16

- Testing technique used to reduce the number of test cases to a manageable level while still maintaining reasonable test coverage.
- Each EC consists of a set of data that is treated the same by the module or that should produce the same result.
- Any data value within a class is *equivalent*, in terms of testing, to any other value.

16

Equivalence Class Testing Technique

17

Partition test cases into classes such that:

1. Every possible input belongs to one of the classes
2. No input belongs to two different classes
3. If we demonstrate a fault in the code for a given input, we should demonstrate a fault with any other input from the same class (with a high probability)
4. Classes are identified by looking at boundary values for the variables of the application

17

Example-1

18

- If (x > y) then S_1 else S_2

Equivalence classes for values of x and y:

18

Example - 2

19

- Taking each input condition (usually a sentence or phrase in the specification) and partitioning it into two or more groups:
- Input condition
  - range of values x: 1-50
- Valid equivalence class
  - ? < x < ?
- Invalid equivalence classes
  - x < ?
  - x > ?

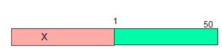
19

Example - 2

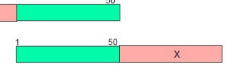
20

- Taking each input condition (usually a sentence or phrase in the specification) and partitioning it into two or more groups:
- Input condition
  - range of values x: 1-50
- Valid equivalence class
  - 0 < x < 51

- Invalid equivalence classes
  - x < 1



  - x > 50



20

Example 3 (Group Exercise)

21

- The program specification states that the program accepts 4 to 10 inputs which are five-digit integers greater than 10,000.

  What are the partitions?

21

Example 3-Partitions

22

The program specification states that the program accepts 4 to 10 inputs which are five-digit integers greater than 10,000.

X < 4	4 ≤ X ≤ 10	X > 10

X = Number of input values

Y <10,000	10,000 ≤ Y ≤ 99,999	Y > 99,999

Y = Actual input value

22

Discussion Point

23

- Consider a module for human resources system that decides how to handle employment process based on the applicant's age. The rules are:
  - 0-16: Don't Hire
  - 16-18: Can hire only as part-time
  - 18-55: Can hire as a full-time
  - 55-99: Don't hire
- Typical test would be (15, Don't Hire)
- How about test cases like (969, …), (FRED, …), (&#$@, …)?

23

Design-by-Contract vs. Defensive Design - 1

24

- Testing based on design-by-contract relies on the module's pre and post conditions
- Precondition: defines what the module requires for it to accomplish its task
  - Example: for a module that reads a file, an appropriate precondition would be that the file is readable
- Post condition: defines the conditions that must be met if the module completes its task(s) successfully
  - Example: file is open and ready for operations
- In testing based on design-by-contract, you have to worry only about testing the conditions satisfied by the precondition

24

Design-by-Contract vs. Defensive Design - 2

25

- In defensive design, the module is designed to accept any input.
  - If preconditions are met, the module will achieve its post conditions
  - Otherwise, the module will notify the caller by returning an error code or throwing an exception
- As a tester, you have to consult with designers in order to understand which approach is being used.
  - Even though they might not be even aware of it

25

Equivalence Class Testing: Guidelines

26

1. If an input condition specifies a *range* of values; identify one valid EC and two invalid EC.
2. If an input condition specifies the *number* (e.g., one through 6 owners can be listed for an automobile); identify one valid EC and two invalid EC (- no owners; - more than 6 owners).
3. If an input condition specifies a set of input values and there is reason to believe that each is handled differently by the program; identify a valid EC for each set and one invalid EC for the other sets.
4. If an input condition specifies a "must be" situation (e.g., first character of the identifier must be a letter); identify one valid EC (it is a letter) and one invalid EC (it is not a letter)
5. If there is any reason to believe that elements in an EC are not handled in an identical manner by the program, split the equivalence class into smaller equivalence classes.

26

Identifying Test Cases

27

- Assign a unique number to each EC.
- Until all valid ECs have been covered by test cases, write a new test case covering as many of the uncovered valid ECs as possible.
- Until all invalid ECs have been covered by test cases, write a test case that cover ***<u>one, and only one</u>***, of the uncovered invalid ECs.

27

Group Exercise

28

- Problem statements: NextDate is a function of three variables: month, day and year. It returns the date of the day after the input date. The month, day and year variables have integer values subject to these conditions:

  C1: 1 <= month <=12

  C2: 1<= day <= 31

  C3: 1812 <= year <= 2012

Valid ECs:

Invalid Ecs:

The craft of testing is to combine test cases to test as many ECs as possible within a single test

But this might cause a problem. Any ideas why?

28

Group Exercise - Answers

29

Valid ECs:

M1 {Month: 1 <= Month <= 12}

D1 {Day: 1 <= Day <= 31}

Y1 {Year: 1812 <= Year <= 2012}

Invalid Ecs:

M2 {Month: 1 > Month },
M3 {Month: Month > 12},
D2 {Day: 1 > Day },
D3 {Day: Day > 31},
Y2 {Year: 1812 > Year },
Y3 {Year: Year > 2012}

Input	Expected Output	Ecs
10/15/2010	10/16/2010	M1, D1, Y1
0/15/2010	Invalid	M2
14/15/2010	Invalid	M3
10/0/2010	Invalid	D2
10/35/2010	Invalid	D3
10/10/1800	Invalid	Y2
10/10/2020	Invalid	Y3

29

Group Exercise

30

Consider the example of the Goofy Mortgage Company (GMC).

- Mortgages are allowed for:
  - A person with income between $1,000 and 83,333
  - Condominiums, Townhouses, Single family dwellings
  - 1 to 5 dwellings in a single mortgage
- Mortgages are not allowed for:
  - Corporations, Trusts, or Partnerships
  - Duplex, Mobile Home, or Tree houses
  - Less then one dwelling or more than 5

Monthly Income	# Dwellings	Applicant	Dwelling Type	Result

30

Group Exercise

31

Consider the example of the Goofy Mortgage Company (GMC).

- Mortgages are allowed for:
  - A person with income between $1,000 and 83,333
  - Condominiums, Townhouses, Single family dwellings
  - 1 to 5 dwellings in a single mortgage
- Mortgages are not allowed for:
  - Corporations, Trusts, or Partnerships
  - Duplex, Mobile Home, or Tree houses
  - Less then one dwelling or more than 5

Monthly Income	# Dwellings	Applicant	Dwelling Type	Result
2500	2	person	townhouse	Grant Mortgage
500	2	person	townhouse	Deny Mortgage
2500	7	person	townhouse	Deny Mortgage
2500	2	Corporation	townhouse	Deny Mortgage
2500	2	Person	Mobile Home	Deny Mortgage

31

Applicability and Limitations

32

- Most suited to systems in which much of the input data takes on values within ranges or within sets.
- It makes the assumption that data in the same EC is, in fact, processed in the same way by the system.
- EC testing is equally applicable at the unit, integration, system, and acceptance test levels. All it requires are inputs or outputs that can be partitioned based on the system's requirements.

32

Boundary Value Testing

33

- Equivalence class testing has the advantage of reducing the number of test case
- But it also has the other advantage of alerting us to the notion of boundary testing
- Boundary value testing focuses on the boundaries simply because that is where so many defects hide. Defects can be in the requirements, design, or code.
- The most efficient way of finding such defects, either in the requirements or the code, is through inspection (Software Inspection, Gilb and Graham's book).

Going back to the HR system example:

0-16:	Don't Hire
16-18:	Can hire only as part-time
18-55:	Can hire as a full-time
55-99:	Don't hire

33

Boundary Value Testing - Technique

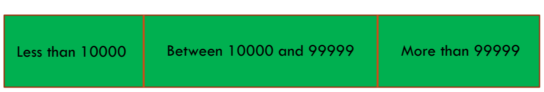
34

- Identify the ECs.
- Identify the boundaries of each EC.
- Create test cases for each boundary value by choosing:
  - one point on the boundary,
  - one point just below the boundary,
  - and one point just above the boundary.

34

Boundary Value Testing - Example

35

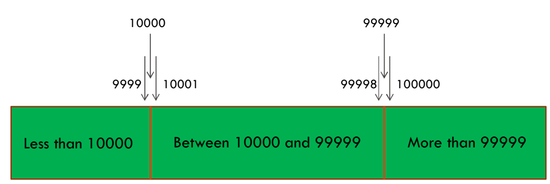


What are the boundary values to test ?

35

Boundary Value Testing - Example

36



What are the boundary values to test ?

36

Boundary Value Analysis Test Cases for Triangle program (1 <= a, b, c <= 200)

37

Test Case	a	b	c	Expected Output
1	100	100	0	Invalid input
2	100	100	1	Isosceles
3	100	100	2	Isosceles
4	100	100	199	Isosceles
5	101	101	200	Isosceles
6	100	100	201	Invalid

37

Group Exercise

38

Given the following method specification:

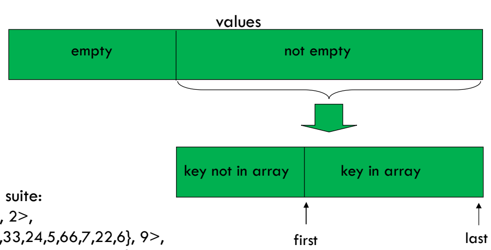
```
/** Returns true iff key is contained in values. */
//@ requires values != null;
//@ ensures \result == (\exist int i; i >= 0 && i < values.length;
//@                         values[i] == key);
public static boolean contains(int[] values, int key)
```

Perform boundary analysis to select a reasonable set of test cases.

38

Answer

39



Test suite:
(<{}, 2>,
<{2,33,24,5,66,7,22,6}, 9>,
<{2,33,24,5,66,7,22,6}, 2>,
<{2,33,24,5,66,7,22,6}, 33>,
<{2,33,24,5,66,7,22,6}, 22>,
<{2,33,24,5,66,7,22,6}, 6>)

39

Applicability and Limitations

40

- Boundary value testing is equally applicable at the unit, integration, system, and acceptance test levels.
- All it requires are inputs that can be partitioned and boundaries that can be identified based on the system's requirements.

40

Decision Table Testing

41

- Decision tables are an excellent tool to capture certain kinds of system requirements and to document internal system design.
- They are used to record complex business rules that a system must implement.
- In addition, they can serve as a guide to creating test cases.

41

Technique: General Format of a Decision Table

42

	Rule 1	Rule 2	...	Rule P
Conditions				
Condition-1				
Condition-2				
...				
Condition-m				
Actions				
Action-1				
Action-2				
...				
Action-n				

42

Don't Care Entries

43

	Rule 1	Rule 2	Rules 3,4	Rule 5	Rule 6	Rules 7,8
C1	T	T	T	F	F	F
C2	T	T	F	T	T	F
C3	T	F	_	T	F	_
A1	X	X		X		
A2	X				X	
A3		X		X		
A4			X			X

The don't care entry has two interpretations:
- the condition is irrelevant, or
- the condition does not apply. Sometimes the "n/a" symbol for this latter interpretation.

43

Types of Decision Tables

44

- Limited entry decision tables:
  - all the conditions are binary
- Extended entry decision tables:
  - conditions are allowed to have several values.!
- Decision tables are deliberately declarative (as opposed to imperative);
  - no particular order is implied by the conditions, and selected actions do not occur in any particular order.
  - Imperative involves spelling out as much detail as necessary how to accomplish something; Declarative involves specifying what needs to be accomplished

44

Decision Table-Based Testing

45

- Applicable for requirements in *if-then*
  - **if** C_1 **and** C_2 **and**…**and** C_n **then** A_k
- Create a decision table comprised of conditions and actions.
- Number of columns: 2^n (n is number of conditions)
- Number of rows: n + m, where m is number of actions
- For each set of conditions, there is a corresponding action

45

Decision Table to Test Cases

46

	Test Case 1	Test Case 2	…	Test Case P
Inputs				
Condition-1				
Condition-2				
…				
Condition-m				
Expected Results				
Action-1				
Action-2				
…				
Action-n				

46

Group Exercise: Triangle program

47

- The program accepts three integers, *a*, *b*, and *c* as input. The three values are interpreted as representing the lengths of sides of a triangle. The integers *a*, *b*, and *c* must satisfy the following conditions:

a < b + c
b < a + c
c < a + b
1<=a<=200
1<=b<=200
1<=b<=200

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Rule 9
C1: a, b, c form a triangle?	N	Y	Y						
C2: a = b?	–	Y	Y						
C3: a = c?	–	Y	Y						
C4: b = c?	–	Y	N						
A1: Not a triangle	X								
A2: Scalene									
A3: Isosceles									
A4: Equilateral		X							
A5: Impossible			X						

- If the integers a, b, and c do not constitute a triangle, we do not even care about possible equalities (rule 1)
- If two pairs of integers are equal, by transitivity, the third pair must be equal; thus, the negative entry (N) makes these rules impossible (rule3).

47

Develop Decision Table for the Triangle Problem

48

- What are the conditions?
  - C1: a, b, c form a triangle?
  - C2: a = b?
  - C3: a = c?
  - C4: b = c?
- What are the actions?
  - Not a triangle, scalene, isosceles, equilateral, impossible
- How many rows?
  - \# of rows = #conditions + #actions
- How many columns?
  - $2^{\#\text{of conditions}}$
  - Or is it?

48

49

	R1	R2	R3	R4	R5	R6	R7	R8	R9
C1: a, b, c form a triangle?	N	Y	Y	Y	Y	Y	Y	Y	Y
C2: a = b?	DC	Y	Y	Y	N	N	Y	N	N
C3: a = c?	DC	Y	Y	N	Y	N	N	Y	N
C4: b = c?	DC	Y	N	Y	Y	N	N	N	Y
A1: Not a Triangle	X								
A2: Scalene						X			
A3: Isosceles							X	X	X
A4: Equilateral		X							
A5: Impossible			X	X	X				

49

Group Exercise: Triangle program

50

- What if we expand the first condition to:

C1: a < b + c ?, C2: b < a + c ?, C3: c < a + b ? What will the decision table look like

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
C1	F	T	T	T	T	T	T	T	T	T	T
C2	-	F	T	T	T	T	T	T	T	T	T
C3	-	-	F	T	T	T	T	T	T	T	T
a=b	-	-	-	T	T	T	T	F	F	F	F
a=c	-	-	-	T	T	F	F	T	T	F	F
b=c	-	-	-	T	F	T	F	T	F	T	F
Not Triangle	X	X	X								
Scalene											X
Isosceles							X		X	X	
Equilateral				X							
Impossible					X	X		X			

50

DT Testing: One More Example

51

- A marketing company wishes to construct a decision table to decide how to treat clients according to three characteristics: Gender, City Dweller, and age group: Young (under 30), Middle-age (between 30 and 60), and Old (over 60). The company has four products (W, X, Y and Z) to test market. Product W will appeal to female city dwellers. Product X will appeal to young females. Product Y will appeal to Male middle-aged shoppers who do not live in cities. Product Z will appeal to all but older females.

51

Solution

52

	1	2	3	4	5	6	7	8	9	10	11	12
Gender	F	M	F	M	F	M	F	M	F	M	F	M
City	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Age	A	A	A	A	B	B	B	B	C	C	C	C
W	X				X				X			
X	X		X									
Y								X				
Z	X	X	X	X	X	X	X	X		X		X

52

Solution (Simplified)

53

	1	2	3	4	5	6	7	8	9	10	11	12
Gender	F	M	F	M	F	M	F	M	F	M	F	M
City	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Age	A	A	A	A	B	B	B	B	C	C	C	C
W	X				X				X			
X	X		X									
Y								X				
Z	X	X	X	X	X	X	X	X		X		X

53

Solution (Simplified)

54

	1	2	3	4	5	7	8	9	11	12
Gender	F	M	F	M	F	F	M	F	F	M
City	Y	Y	N	N	Y	N	N	Y	N	N
Age	A	DC	A	A	B	B	B	C	C	C
W	X				X			X		
X	X		X							
Y							X			
Z	X	X	X	X	X	X	X			X

54

Advantages/Disadvantages of Decision Table

55

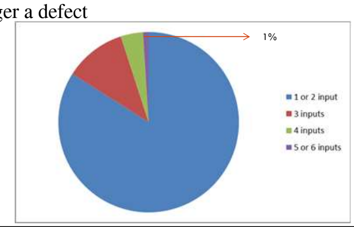
- Advantages: (check <u>completeness</u> & <u>consistency</u>)
  1. Allow us to start with a "complete" view, with no consideration of dependence
  2. Allow us to look at and consider "dependence," "impossible," and "not relevant" situations and eliminate some test cases.
  3. Allow us to detect potential error in our Specifications
- Disadvantages:
  1. Need to decide (or know) what conditions are relevant for testing - - - this <u>may require Domain knowledge</u>
     - e.g. need to know leap year for "next date" problem in the book
  2. Scaling up can be massive: <u>2^n rules for n conditions - - - that's if the conditions are binary and gets worse if the values are more than binary</u>

55

Pairwise Testing

56

- Test all pairs of variables when combinations to test is very large
- Most defects are either ***single-mode*** or ***double-mode*** defects
  - Single-mode – Defects caused by specific states or values of a single variable
  - Double-mode – Defects that occur when a combination of two variables assume specific states or values
- # of tests needed to trigger a defect



56

Pairwise Testing

57

- Test all pairs of variables when combinations to test is very large
- Most defects are either ***single-mode*** or ***double-mode*** defects
- Pairwise testing defines a minimal subset that guides us to test for all single-mode and double-mode defects
- Four parameters, with three possible values each
  - $3^4 = 81$ possibilities
  - Can be done in 9 tests
- 13 parameters, with three possible values each
  - $3^{13} = 1{,}594{,}323$ possibilities
  - Can be done in 15 tests
- 20 parameters, with 10 possible values each
  - 10^{20} = MANY possibilities
  - Can be done in 180 tests

57

Orthogonal Arrays

58

- Two-dimensional array of numbers that has this property:
  - Choose any two columns in the array, all the pairwise combinations of values will occur in within the two columns.
- Three variables, with two possible values each:
  - $2^3 = 8$ Possibilities

	A	B	C
1	1	1	1
2	1	1	2
3	1	2	1
4	1	2	2
5	2	1	1
6	2	1	2
7	2	2	1
8	2	2	2

58

Orthogonal Arrays

59

- Two-dimensional array of numbers that has this property:
  - Choose any two columns in the array, all the pairwise combinations of its values will occur in every pair of columns.
  - If a pair occur in the array multiple times, then all other pairs will occur the same number of times
- Three variables, with two possible values each:
  - $2^3 = 8$ Possibilities, but we could do this in half the test cases

	A	B	C
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

Note that we are not looking for triplets such as 121. Only pairs

59

Orthogonal Array Example

60

- An orthogonal array with 4 columns (attributes), maximum value 3 (1-3)
- $L_9(3^4)$

# of columns

# of rows

Maximum value

	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

60

Orthogonal Arrays

61

- Not all columns must have the same range of values
  - $L_{18}(2^1 3^7)$: 18 rows, 1 column of 1s and 2s, 7 columns of 1s, 2s, and 3s
  - Good news about pairwise testing is that we do not actually have to come up with the OA.
  - Tools will assist in doing this task
    - http://designcomputing.net/gendex/noa/
    - https://app.hexawise.com
    - https://inductive.no/pairwiser-tool/

61

Using Orthogonal Arrays

62

a) Identify the variables
b) Determine the number of choices for each variable
c) Locate an orthogonal array which has a column for each variable and values within the columns that correspond to the choices for each variable
d) Map the test problem onto the orthogonal array
e) Construct the test cases

A library of over 200 orthogonal arrays:
http:/www.research.att.com/~njas/oadir/
Maintained by N.J. A. Sloane.

62

Example: Mortgage Application

63

- Loan Details
  - Term of loan (30yrs, 20yrs, 15yrs)
  - Loan Amount (Large, Medium, Small)
  - LTV Ratio (90%, 80%, 70%)

27 possibilities

- Customer Details
  - Income (High, Medium, Low)
  - Credit Score (High, Medium, Low)
  - Customer Status (VIP, Regular, Employee)

27 possibilities

- Property Details
  - Type (House, Apartment, Condo)
  - Location (Washington, California, NY)
  - Residence Status (Primary, Rent, Investment)

27 possibilities

Variables and Possible values

All Possibilities? 27+27+27? 27*27*27? 19683

63

In your teams

64

- Create an account in https://inductive.no/
- Create an orthogonal array for the previous problem
- How many test cases do you need?

64

Example: Mortgage Application

65

	term	amount	ratio	Income	C_Score	C_Status	Type	Loc	R_Status
1	30	large	70	L	H	Reg	House	CA	Rent
2	15	small	70	M	L	Reg	House	NY	Prim
3	20	small	80	L	H	Emp	Condo	CA	Prim
4	15	small	70	H	M	VIP	Condo	NY	Invst.
5	20	medium	80	L	L	Reg	Appt.	NY	Rent
6	30	large	80	H	H	Emp	Appt.	NY	Invst.
7	15	medium	80	L	L	VIP	Condo	CA	Prim
8	20	large	90	M	L	Reg	Appt.	CA	Invst.
9	30	small	90	M	H	Emp	Appt.	WA	Rent
10	15	medium	80	L	H	Emp	Condo	WA	Prim
11	30	medium	90	H	M	VIP	House	WA	Rent
12	20	medium	70	L	M	Emp	Appt.	CA	Invst.
13	20	large	80	H	M	VIP	Condo	CA	Prim
14	15	large	80	M	M	Reg	Condo	WA	Rent
15	30	large	80	H	L	Emp	House	NY	Prim
16	20	medium	70	M	L	VIP	Appt.	WA	Invst.
17	15	small	90	H	H	Reg	Appt.	NY	Prim
18	30	small	90	L	H	VIP	Condo	CA	Prim
19	20	large	80	M	H	Emp	House	NY	Invst.

65

Example: Mortgage Application

66

e) Construct the test cases

- Construct a test case for each row in the orthogonal array
- 19 test cases
- Any single-mode or double mode defect will be discovered

66

Applicability and Limitations

67

- It is equally applicable at the unit, integration, system and acceptance test levels
- Reduces the number of test combination significantly
- Issues with OA use

67