SE-3002
SOFTWARE QUALITY ENGINEERING

RUBAB JAFFAR
RUBAB.JAFFAR@NU.EDU.PK

Pert II-Software Testing

Functional Testing

Lecture # 13, 14, 15
04, 05, 07 Oct

TODAY'S OUTLINE

- Functional testing
- Boundary value analysis
- Equivalence class testing
- Decision table based testing
- Revision

EQUIVALENCE CLASS TESTING

- A large number of test cases are generated for any program. It is neither feasible nor desirable to execute all such test cases.
- Select a few test cases and still wish to achieve a reasonable level of coverage.
- Divide input domain into various categories with some relationship and expect that every test case from a category exhibits the same behavior.
- If categories are well selected, we may assume that if one representative test case works correctly, others may also give the same results. This assumption allows us to select exactly one test case from each category and if there are four categories, four test cases may be selected.
- Each category is called an equivalence class and this type of testing is known as equivalence class testing.

CREATION OF EQUIVALENCE CLASSES

- The entire input domain can be divided into at least two equivalence classes: one containing all valid inputs and the other containing all invalid inputs.
- Each equivalence class can further be sub-divided into equivalence classes on which the program is required to behave differently.
- Square program

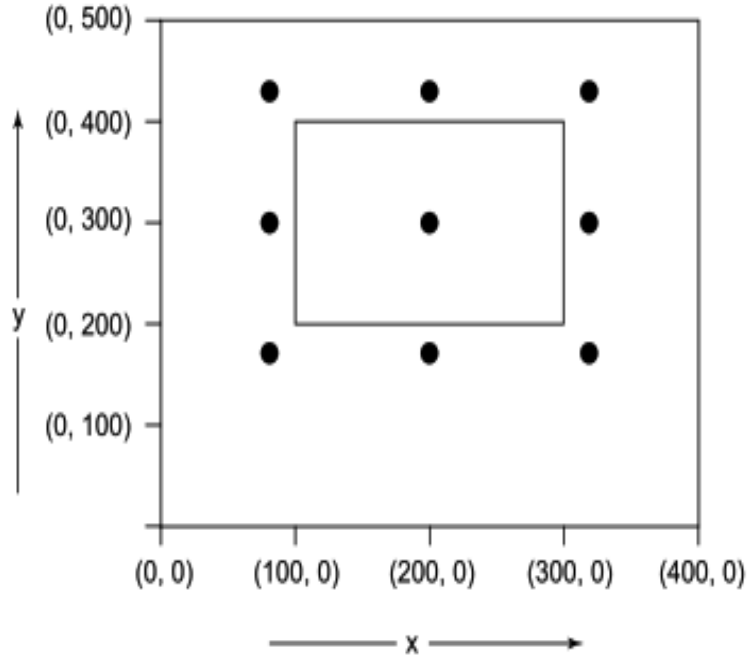
(i) $I_1 = \{ 1 \leq x \leq 100 \}$ (Valid input range from 1 to 100)
(ii) $I_2 = \{ x < 1 \}$ (Any invalid input where x is less than 1)
(iii) $I_3 = \{ x > 100 \}$ (Any invalid input where x is greater than 100)

Test cases for program 'Square' based on input domain

Test Case	Input x	Expected Output
I_1	0	Invalid Input
I_2	50	2500
I_3	101	Invalid Input

EQUIVALENCE CLASSES FOR ADDITION PROGRAM

(i) $I_1 = \{ 100 \leq x \leq 300 \text{ and } 200 \leq y \leq 400 \}$ (Both x and y are valid values)
(ii) $I_2 = \{ 100 \leq x \leq 300 \text{ and } y < 200 \}$ (x is valid and y is invalid)
(iii) $I_3 = \{ 100 \leq x \leq 300 \text{ and } y > 400 \}$ (x is valid and y is invalid)
(iv) $I_4 = \{ x < 100 \text{ and } 200 \leq y \leq 400 \}$ (x is invalid and y is valid)
(v) $I_5 = \{ x > 300 \text{ and } 200 \leq y \leq 400 \}$ (x is invalid and y is valid)
(vi) $I_6 = \{ x < 100 \text{ and } y < 200 \}$ (Both inputs are invalid)
(vii) $I_7 = \{ x < 100 \text{ and } y > 400 \}$ (Both inputs are invalid)
(viii) $I_8 = \{ x > 300 \text{ and } y < 200 \}$ (Both inputs are invalid)
(ix) $I_9 = \{ x > 300 \text{ and } y > 400 \}$ (Both inputs are invalid)



TEST CASES FOR ADDITION

Test cases for the program 'Addition'

Test Case	x	y	Expected Output
I_1	200	300	500
I_2	200	199	Invalid input
I^3	200	401	Invalid input
I_4	99	300	Invalid input
I_5	301	300	Invalid input
I_6	99	199	Invalid input
I_7	99	401	Invalid input
I_8	301	199	Invalid input
I_9	301	401	Invalid input

APPLICABILITY

- Applicable at unit, integration, system and acceptance test levels.
- The basic requirement is that inputs or outputs must be partitioned based on the requirements and every partition will give a test case.
- If one test case catches a bug, the other probably will too. If one test case does not find a bug, the other test cases of the same equivalence class may also not find any bug.
- The design of equivalence classes is subjective and two testing persons may design two different sets of partitions of input and output domains.

EXAMPLE

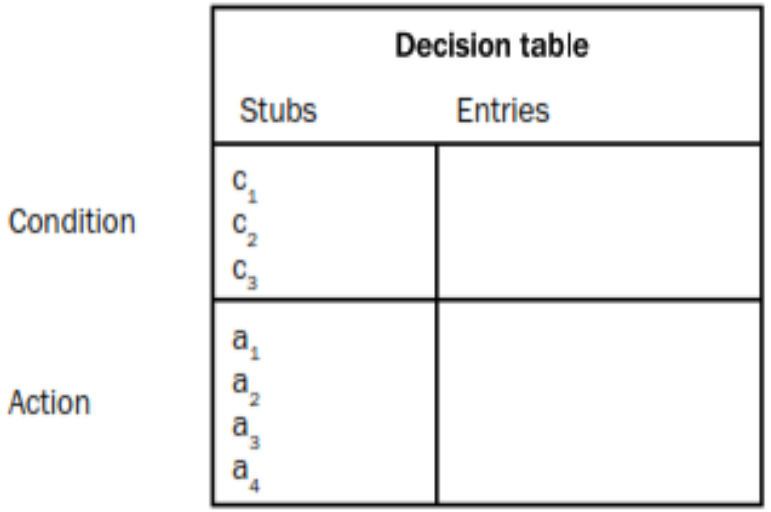
- Consider a program for the determination of the largest amongst three numbers. Its input is a triple of positive integers (say x, y and z) and values are from interval [1, 300].
- Design the boundary value test cases.
- Design the equivalence classes

DECISION TABLE BASED TESTING

- In Software Engineering, boundary value and equivalent partition are other similar techniques used to ensure better coverage.
- They are used if the system shows the **same** behavior for a large set of inputs.
- However, in a system where for each set of input values the system behavior is **different**, boundary value and equivalent partitioning technique are not effective in ensuring good test coverage.
- In this case, decision table testing is a good option. This technique can make sure of good coverage, and the representation is simple so that it is easy to interpret and use.
- This table can be used as the reference for the requirement and for the functionality development since it is easy to understand and cover all the combinations.

DECISION TABLE BASED TESTING

- Decision tables are used in many engineering disciplines to represent complex logical relationships.
- An output may be dependent on many input conditions and decision tables give a pictorial view of various combinations of input conditions.
- There are four portions of the decision table. The decision table provides a set of conditions and their corresponding actions.



Four Portions

1. Condition Stubs
2. Condition Entries
3. Action Stubs
4. Action Entries

PARTS OF THE DECISION TABLE

- The four parts of the decision table are given as:
- Condition Stubs: All the conditions are represented in this upper left section of the decision table. These conditions are used to determine a particular action or set of actions.
- Action Stubs: All possible actions are listed in this lower left portion of the decision table.
- Condition Entries: In the condition entries portion of the decision table, we have a number of columns and each column represents a rule. Values entered in this upper right portion of the table are known as inputs.
- Action Entries: Each entry in the action entries portion has some associated action or set of actions in this lower right portion of the table. These values are known as outputs and are dependent upon the functionality of the program.

TYPICAL STRUCTURE OF DECISION TABLE

Typical structure of a decision table

Stubs	R_1	R_2	R_3	R_4
c_1	F	T	T	T
c_2	-	F	T	T
c_3	-	-	F	T
a_1	X	X		X
a_2			X	
a_3	X			

DECISION TABLE FOR A LOGIN SCREEN.

- The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.
- Case 1 – Username and password both were wrong. The user is shown an error message.
- Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
- Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
- Case 4 – Username and password both were correct, and the user navigated to homepage

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	F	T	F	T
Password (T/F)	F	F	T	T
Output (E/H)	E	E	E	H

EXAMPLE 2: DECISION TABLE FOR UPLOAD SCREEN

- Develop a decision table for an application that allows a user to upload an image. For uploading an image, application opens a dialogue box which will ask the user to upload photo with certain conditions like –
- You can upload only '.jpg' format image
- file size less than 32kb
- resolution 137*177.

DECISION TABLE FOR UPLOAD SCREEN

Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Format	.jpg	.jpg	.jpg	.jpg	Not .jpg	Not .jpg	Not .jpg	Not .jpg
Size	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb
resolution	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177
Output	Photo uploaded	Error message resolution mismatch	Error message size mismatch	Error message size and resolution mismatch	Error message for format mismatch	Error message format and resolution mismatch	Error message for format and size mismatch	Error message for format, size, and resolution mismatch

APPLICABILITY

- Decision tables are popular in circumstances where an output is dependent on many conditions and a large number of decisions are required to be taken.
- They may also incorporate complex business rules and use them to design test cases.
- Every column of the decision table generates a test case.
- As the size of the program increases, handling of decision tables becomes difficult and cumbersome.
- In practice, they can be applied easily at unit level only. System testing and integration testing may not find its effective applications.

APPLICABILITY

- It's a tabular representation of input conditions and resulting actions. Additionally, it shows the causes and effects. Therefore, this technique is also called a ***cause-effect table*.**
- Testing combinations can be a challenge, especially if the number of combinations is enormous. Moreover, testing all combinations is not practically feasible as it's not cost and time effective. Therefore, we have to be satisfied with testing just a small subset of combinations. That is to say, the success of this technique depends on our choice of combinations

That is all