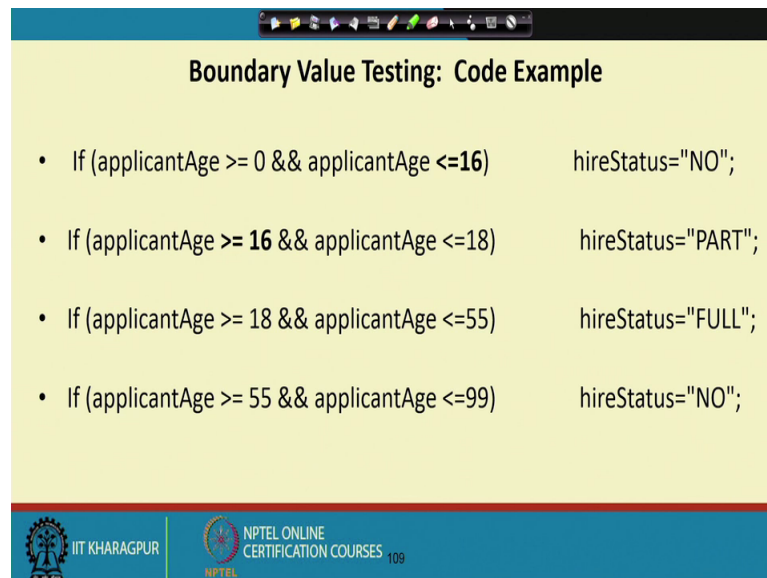


Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 50
Special Value Testing



Welcome to this lecture. In the last lecture, we were discussing about a black box testing technique known as the boundary value testing. And we had said that, this is a general risk; the boundaries are at general risk of errors and therefore, we must define test cases based on the boundary values. Let us proceed from what we were discussing last time.

(Refer Slide Time: 00:48)



Boundary Value Testing: Code Example

- If (applicantAge >= 0 && applicantAge <=16) hireStatus="NO";
- If (applicantAge >= 16 && applicantAge <=18) hireStatus="PART";
- If (applicantAge >= 18 && applicantAge <=55) hireStatus="FULL";
- If (applicantAge >= 55 && applicantAge <=99) hireStatus="NO";

 IIT KHARAGPUR  NPTEL ONLINE
CERTIFICATION COURSES 109

Let us say, we have a code were depending on the applicant age, we make the hire status; it is either no, do not hire under age or hire status is part time full time and over age, hire status is no. We can identify the boundaries here which is between these two equivalence classes, 16 is the boundary and if we select the boundary likely to determine the bug here that 16 is included in both the equivalence classes, similarly 18, 55 and so on.

(Refer Slide Time: 01:42)

Boundary Value Testing Example (cont)

- Corrected boundaries:

0–15	Don't hire
16–17	Can hire on a part-time basis only
18–54	Can hire as full-time employees
55–99	Don't hire
- What about ages -3 and 101?
- The requirements do not specify how these values should be treated.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 110

And then, we can correct our code where we have the boundaries clearly specified here 0 to 15, do not hire 16 to 17, 18 to 54 and so on. And then, we can take care of values that are invalid and we need to take care of the boundary between the invalid and the valid set of equivalence classes.

(Refer Slide Time: 02:16)

Boundary Value Testing Example (cont)

- The code to implement the corrected rules is:

```
if (applicantAge >= 0 && applicantAge <=15) hireStatus="NO";  
if (applicantAge >= 16 && applicantAge <=17) hireStatus="PART";  
if (applicantAge >= 18 && applicantAge <=54) hireStatus="FULL";  
if (applicantAge >= 55 && applicantAge <=99) hireStatus="NO";
```
- Special values on or near the boundaries in this example are {-1, 0, 1}, {14, 15, 16}, {17, 18, 19}, {54, 55, 56}, and {98, 99, 100}.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 111

So, based on the boundary values, we have the corrected code and then what if for this code, we have let us say minus 1. That is not addressed here in the code. And therefore, it will have generate a failure. Depending on the 4 equivalence classes here, we can define

the boundary values here; for the first one -1, 0, 1, 14, 15 and 16; for the second one, 17, 18, 19; 17 at the boundary of this. So, there are basically if we look at here, we have one boundary that is less than 0, another boundary here at 15 and 16 and another here at 17 and another here at 54 and so on. We must include the boundaries here for each boundary we take a test case which is just on the boundary one, on the higher side and lower side and so on.

So, these are each of this either test case.

(Refer Slide Time: 03:56)

The slide is titled "Boundary Value Analysis" and contains a bullet point: "Create test cases to test boundaries of equivalence classes". Below the text is a diagram of a complex, irregular shape divided into several regions by a central vertical line and other internal lines. Red circles are drawn around the boundaries of these regions, indicating the focus of boundary value analysis. The slide also features a small video inset of a speaker in the bottom right corner and logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES 112 at the bottom.

And once we identify the boundary values, then we need to define the test cases. These are the valid set of equivalence classes, invalid set of equivalence classes. And we already had selected one representative value for the equivalence class testing. Now, we will select values in the boundary for the boundary value testing.

(Refer Slide Time: 04:21)

Example 1

- For a function that computes the square root of an integer in the range of 1 and 5000:
 - Test cases must include the values:
 $\{0, 1, 2, 4999, 5000, 5001\}$.

The diagram shows a number line from 0 to 5000. The region from 0 to 1 is labeled 'Invalid'. The region from 1 to 5000 is labeled 'valid'. The region from 5000 to 5001 is labeled 'Invalid'. The value 1 is circled, and the value 5000 is also circled.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 113

Let us look at some example problems, let take the same problem that we are considering earlier for equivalence class testing that it takes one parameter only; the valid values are between 1 and 5000. If we draw this, we represent the data then we have one valid class and two invalid classes. And there are two boundaries here; one is between the valid class and the invalid on the lower side and another is between the valid class and the invalid on the upper side. We include the boundary 1 and one element, one test case which is less than 1 which is 0, 1, one element which is valid. So, it can be 2. Similarly 5001, 5000 and let us say 4999, 4 9 9 9.

So, these are the set of test cases for one parameter which is a range of values between 1 and 5000.

(Refer Slide Time: 05:46)

• Consider a program that reads the “age” of employees and computes the average age.

input (ages) → Program → output: average age

Assume valid age is 1 to 150

• How would you test this?

- How many test cases would you generate?
- What type of test data would you input to test this program?

Example 2

At least 5 test case

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 114

Now, let us look at age as a parameter and the valid age range is 1 to 150. Here, we have three equivalence classes; one is a valid equivalence class which is between 1 and 150 and there are two equivalence classes. So, for boundary value testing, how many test cases do we need? Of course, need to include the boundary, the boundary value, one below the boundary value, above the boundary value and of course, we can combine these two, we can select a value anywhere here. So, there will be at least 5 test cases, at least 5 test cases for boundary value testing.

(Refer Slide Time: 07:02)

Boundaries of the inputs

The “basic” boundary value testing would include 5 test cases:

1. - at minimum boundary
2. - immediately above minimum
3. - between minimum and maximum (nominal)
4. - immediately below maximum
5. - at maximum boundary

predict-longevity(age)

$1 \leq \text{age} \leq 150$

1 ← age → 150

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 115

So, the 5 test cases: at the minimum boundary, immediately above the minimum, between the minimum and maximum, immediately below the maximum and at the maximum boundary.

(Refer Slide Time: 07:21)

Test Cases for the Example

- How many test cases for the example ?
 - answer : **5**
- Test input values? :
 - 1 at the minimum
 - 2 at one above minimum
 - 45 at middle
 - 149 at one below maximum
 - 150 at maximum

`predict-longevity(age)`

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 116

Just an example predict longevity and age is the parameter and age is between 100 and 150 and once we identify the 5 boundaries, sorry 5 boundary values, you can represent them 1, 2, 45, 149 and 150 and this form our boundary value test cases.

(Refer Slide Time: 07:54)

Multiple Parameters: Independent distinct Data

- Suppose there are 2 “distinct” inputs that are assumed to be independent of each other.
 - Input field 1: years of education (say 1 to 23)
 - Input field 2: age (1 to 150)
- If they are independent of each other, then we can start with $5 + 5 = 10$ sets.

coverage of input data: yrs of ed

1. n = 1; age = whatever(37)
2. n = 2; age = whatever
3. n = 12; age = whatever
4. n = 22; age = whatever
5. n = 23; age = whatever

$m+n-1$

coverage of input data: age

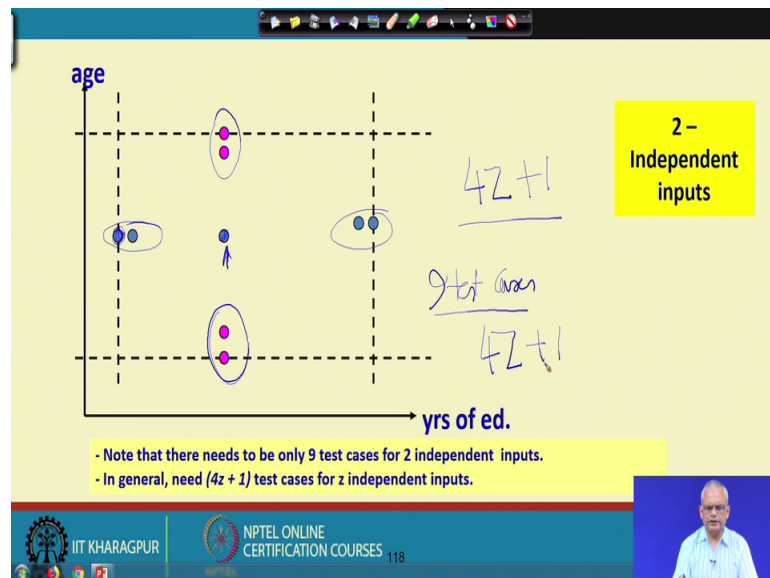
1. n = 12; age = 1
2. n = 12; age = 2
3. n = 12; age = 37
4. n = 12; age = 149
5. n = 12; age = 150

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 117

But, what about multiple parameters; for each of the parameter, we might identify different boundaries and let us say for parameter years of education and let us say, age we identify different boundaries. And let us say the boundary value for the years of education is 1, 2, 12, 22, 23; these are the boundary values for years of education. Similarly, 1, 2 sorry for the age, we have 1, 2, 37, 149 and 50. These are boundary values. Now, how do we define the test cases by combining these two boundaries? One way is that we just represent each of this in the test case and take any one of the value from the other equivalence class for the other parameter. And similarly, we define test cases corresponding to each of the boundaries here and some value of the other boundaries; pick one boundary from here, the first parameter and combine with all other parameter boundaries here.

And similarly, for every value on the boundary and the first parameter, we take any value from the second parameter like 37 or something. So, what will be the number of test cases that will be required? If there are let us say m here, m boundary values for the first parameter and n boundary values for the second parameter, we will have m plus n minus 1 because 1 is common here. Therefore, we will have m plus n minus 1.

(Refer Slide Time: 10:31)



And we represent the same thing here. This is called, this is for 2 input parameters and we define the boundaries here. Let us say for the first parameter, we define there are 5 boundary values and for the second parameter also, there are 5 boundary values and this

is common here. For the first parameter, these two are the boundary and this is a point just inside and these are the two points on the boundary and this is just inside. Now, if we combine these two parameters, we take this one, this one, this one, these two; and therefore, we get 9 test cases.

And we can represent it as $4z$ where z is the number of parameters plus 1 because for every boundary, we have 4 and one representative value here. And similarly for the other one, for the boundary and one representative value, and therefore, we have $4z$ plus 1 which is the number of test cases required.

(Refer Slide Time: 12:25)

Boundary Value Test

Given $f(x,y)$ with constraints \rightarrow

$$a \leq x \leq b$$

$$c \leq y \leq d$$

Boundary Value analysis focuses on the boundary of the input space to identify test cases.

Defined as input variable value at min, just above min, a nominal value, just above max, and at max.

$4 * 2 + 1 = 9$

The slide includes a 2D plot with x and y axes. The x-axis has points 'a' and 'b', and the y-axis has points 'c' and 'd'. Dashed lines form a rectangle. Test cases are marked with red dots: one at each corner (a,b), (a,c), (a,d), (b,c), (b,d), one at the midpoint of each side, and one in the center. A small video inset of a speaker is in the bottom right corner.

Let us say, we have a function $f(x,y)$, the function f takes two parameters, two integer parameters x and y and x takes value between a and b and y takes value between c and d now how many test cases are needed. If we represent the boundary here, we have a boundary at c , we have boundary at d , we have boundary at a and b and just above we have test cases and one representative here. So, it will be 4 into 2 parameters plus 1 which is equal to 9 . There are 2 parameters here and each one, we need 4 and 1 is the common and therefore, we need 9 .

(Refer Slide Time: 13:28)

Weak Testing: Single Fault Assumption

- **Premise:** “Failures rarely occur as the result of the simultaneous occurrence of two (or more) faults”
- Under this:
 - Hold the values of all but one variable at their nominal values, and let that one variable assume its extreme values.

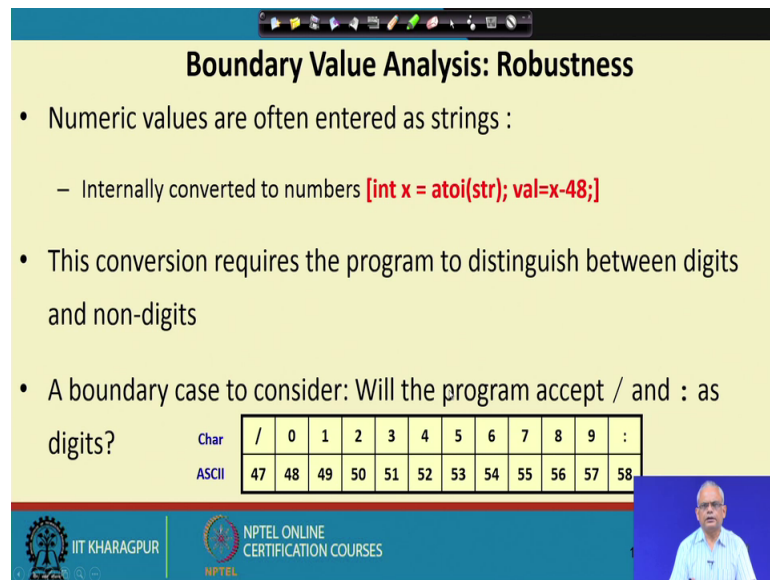
The slide features a 3x3 grid diagram with handwritten labels P_1 on the left and P_2 at the bottom. The top-middle cell contains a circled 'X', and the middle-right cell contains an asterisk '*'. The bottom-left cell contains a circled 'X'. The slide footer includes the IIT KHARAGPUR logo, NPTEL ONLINE CERTIFICATION COURSES 120, and a small video of the presenter.

All the weak testing that we have been discussing with respect to equivalence class multiple parameter equivalence class and boundary value, the weak testing actually implicitly assumes a single fault that is only one of the parameter can have fault. If that is assumption, then weak testing is alright because for every input, every representative of one parameter, we combine with the representative for the other parameter.

Therefore, if this is the let me just explain this, that if we have two parameters for weak testing if you remember, if there are let us say 3 equivalence classes for the first parameter and 3 equivalence classes for the second parameter, then we may select this and this. So, we have all the parameters of the all the representatives for the parameter 1 and parameter 2 represented in our test cases. And therefore, if there is a fault in any parameter representative parameter that will be detected, but what if specific combinations of parameters have fault.

Let us say when parameter 1 has representative from this and the corresponding one is here, if these two are chosen then there is a fault. Of course, weak testing will not be able to detect that. It will detect only when one of the parameter, any one of the parameter if we select a value there, then it is faulty. In those cases, weak testing is satisfactory; otherwise we need to go for strong testing.

(Refer Slide Time: 15:48)



Boundary Value Analysis: Robustness

- Numeric values are often entered as strings :
 - Internally converted to numbers `[int x = atoi(str); val=x-48;]`
- This conversion requires the program to distinguish between digits and non-digits
- A boundary case to consider: Will the program accept / and : as digits?

Char	/	0	1	2	3	4	5	6	7	8	9	:
ASCII	47	48	49	50	51	52	53	54	55	56	57	58

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us look at the robust testing in robust testing we need to include a value which is invalid. Let us say, we have a function called as ASCII to integer and it takes a string let us say character it takes and it gives us a integer value for that. And if we look at the ASCII table, that 0 to 9 are the characters that are valid correct which are the valid characters for integers and they have values 48 to 57. But then, if we give a invalid value such as slash or colon, then we can check that it gives us a wrong value. And therefore, we must include the one which is invalid integer representation.

And that we call as the robustness testing.

(Refer Slide Time: 17:03)

Robustness testing

- This is just an extension of the Boundary Values to include invalid values:
 - Less than minimum
 - Greater than maximum
- There are 7 test cases for each input

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 122

For robustness testing therefore, we consider the invalid values as well. And therefore, the number of elements at the boundary are three on each of the boundary. For x sorry the parameter 1, we have 2 boundaries and for both boundaries we need to select three element each. And similarly for parameter 2 sorry, for parameter 2, we need to select three elements each for parameter one we also select 3 element each and then this is the general element with which we combine and therefore, we need 7 test cases for each input; if we have n test cases, sorry if we have n parameters.

(Refer Slide Time: 18:19)

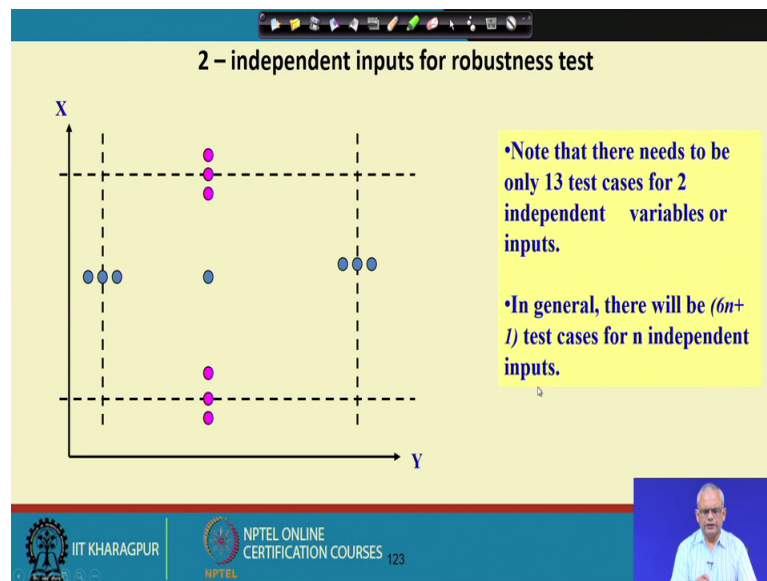
Robustness testing

- This is just an extension of the Boundary Values to include invalid values:
 - Less than minimum
 - Greater than maximum
- There are 7 test cases for each input
- The testing of robustness is really a test of "error" handling.
 1. Did we anticipate the error situations?
 2. Do we issue informative error messages?
 3. Do we support some kind of recovery from the error?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 122

Then we can easily compute how many test cases we need. The main idea behind robustness testing is that do we anticipate the error situations, do we display informative error messages when the value is invalid is there, some kind of recovery from the error. So, those are the main objectives of the robustness testing to check whether the programmer has handled invalid value satisfactorily.

(Refer Slide Time: 18:50)



By considering invalid values as well, we see that for two input, two parameters we need 13 test cases; so, 6 plus 6 plus 1, 13 test cases. So, in general if we have n parameters, we need $6n + 1$ test cases.

(Refer Slide Time: 19:24)

Some Limitations of Boundary Value Testing

- How to handle a set of values?
- How about set of Boolean variables?
 - True
 - False
- May be radio buttons
- What about a non-numerical variable whose values are text?

Select a size for pizza

Small

Medium

Large

CANCEL OK

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 124

But the main problem with boundary value testing is that if we have a set of values, we do not have boundary there. What about Boolean values, there also we do not have boundaries; what about radio buttons like this, these can be considered like Boolean variables either they are on or off. Here also, we do not have boundary for this cannot be defined. What about a non numerical parameter, let us say character string?

Again, it is hard to define boundaries for that.

(Refer Slide Time: 20:13)

Quiz: BB Test Design

- Design black box test suite for a function that solves a quadratic equation of the form $ax^2+bx+c=0$.
- Equivalence classes
 - Invalid Equation
 - Valid Equation: Roots?

quadratic solver (a,b,c) {

Valid → Complex

Valid → Real → Coincident

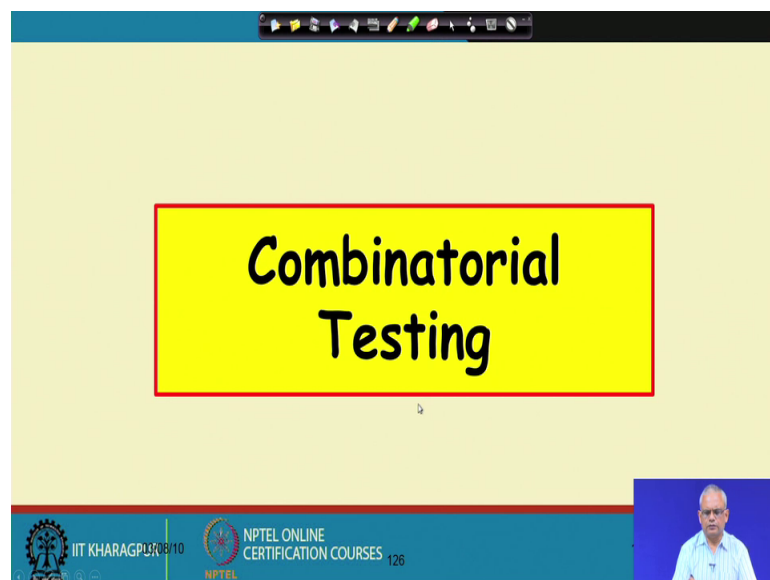
Valid → Real → Unique

3

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 125

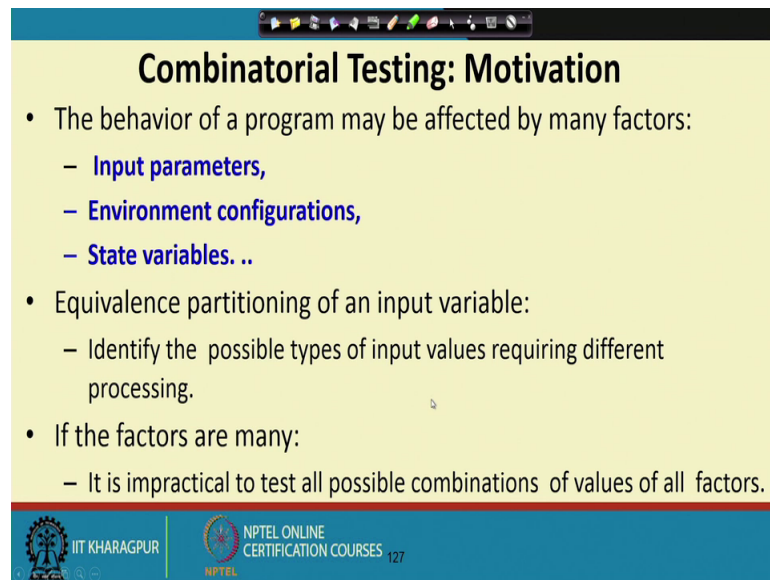
Now, let us conclude our discussion on the equivalence class and boundary value testing, but before that let us do a small quiz. Let us say we have a function which solves the quadratic equation of the form $a x^2 + b x + c$. The name of the function let us say, quad solver and it takes 3 parameters a, b, c and it returns the roots. What will be the equivalence classes for this? If we examine the scenarios here, then one is that the root may be complex, the root may be real. If it is real, it may be coincident or unique the valid set of values can be classified, the equivalence classes will be for complex root, real root and the real root can be coincident and unique and we can also have the invalid set of values and so on.

(Refer Slide Time: 21:55)



Now, let us look at Combinatorial Testing. We will see that if the number of parameters to a function is large and it uses many global variables and so on, the number of test cases required for equivalence class testing and boundary values testing can be large. And we will see how to handle that problem and let us look at combinatorial testing.

(Refer Slide Time: 22:28)



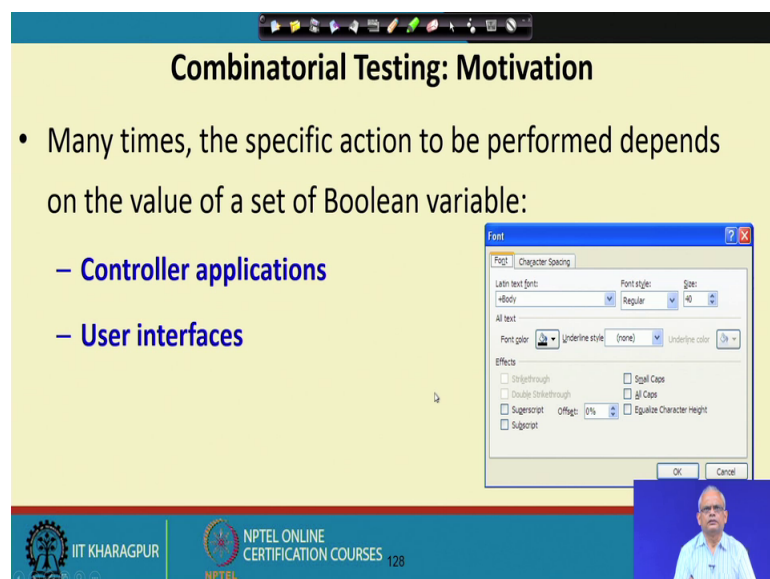
Combinatorial Testing: Motivation

- The behavior of a program may be affected by many factors:
 - **Input parameters,**
 - **Environment configurations,**
 - **State variables. ..**
- Equivalence partitioning of an input variable:
 - Identify the possible types of input values requiring different processing.
- If the factors are many:
 - It is impractical to test all possible combinations of values of all factors.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 127

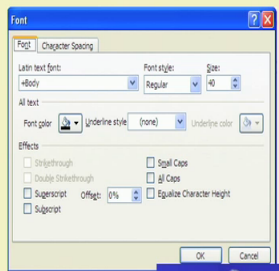
The behavior of a program is affected by many factors; how many input parameters are there, how many global variables are there, state variables and so on. And for equivalence partitioning, we identify for every parameter what are the equivalence classes. And when we combine it, we saw that there is a weak testing, strong testing and robust testing. And if we do a robust testing then it may become impractical, if the number of input parameters are many.

(Refer Slide Time: 23:16)



Combinatorial Testing: Motivation

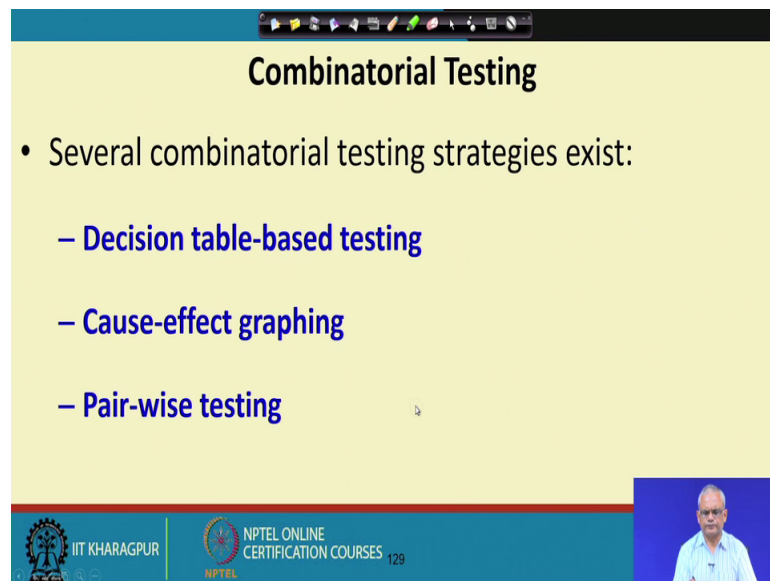
- Many times, the specific action to be performed depends on the value of a set of Boolean variable:
 - **Controller applications**
 - **User interfaces**



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 128

But then, in many situations we find that the number of input parameters are actually too many specially in Controller applications and User interfaces. Let us look at this specific user interface just see how many parameters are there input parameters they are just too many and if we combine the equivalence classes here, then we will get too many test cases.

(Refer Slide Time: 23:52)



The slide is titled "Combinatorial Testing" and lists three strategies:

- Several combinatorial testing strategies exist:
 - **Decision table-based testing**
 - **Cause-effect graphing**
 - **Pair-wise testing**

The slide also features a small video inset of a speaker in the bottom right corner and a footer with logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES 129.



For combinatorial testing, we will see the Decision table-based testing, Cause-effect graphing and Pair-wise testing which will help us to handle a small number of test cases when we have many parameters or many different types of inputs.

(Refer Slide Time: 24:19)

- Applicable to requirements involving conditional actions.
- This is represented as a decision table:
 - Conditions = inputs
 - Actions = outputs
 - Rules =test cases
- Assume independence of inputs
- Example
 - If c1 AND c2 OR c3 then A1

Decision table-based Testing (DTT)

	Rule1	Rule2	Rule3	Rule4
Condition1	Yes	Yes	No	No
Condition2	Yes	X	No	X
Condition3	No	Yes	No	X
Condition4	No	Yes	No	Yes
Action1	Yes	Yes	No	No
Action2	No	No	Yes	No
Action3	No	No		





First let us look at the Decision table based testing. The decision table based testing is applicable when there are many conditional actions here we develop a table and in the table we represent the conditions, the conditions represent inputs and then the actions. The actions represent the output and each of these column under table is called as a rule and the rules are the test cases. Let us see how to generate the decision table and then once we generate the decision table we can easily form the test cases. Each of the column will become a test case.

(Refer Slide Time: 25:15)

- Applicable to requirements involving conditional actions.
- This is represented as a decision table:
 - Conditions = inputs
 - Actions = outputs
 - Rules =test cases
- Assume independence of inputs
- Example
 - If c1 AND c2 OR c3 then A1

Decision table-based Testing (DTT)

Condition	c1	T	F	F	F
	c2	T	T	F	F
	c3	T	T	T	F
Action		A	A	A	-



Let us take the example of this condition here.

C_1 and c_2 or c_3 , then A_1 we can observe here that c_1, c_2, c_3 are the input and A_1 is the output. We write here the input c_1, c_2, c_3 these we call as the condition and then we have the action. And we combine all possible values of the condition with action and let us say the condition is for c_1 , it is true, c_2 is true and c_3 is true; c_1, c_2, c_3 are all true and then the action is A_1 . We can look at the expression here. Now what if c_1 is false, c_2 is true and c_3 is true. In this case, it is also A_1 now what if c_1 is false c_2 is false and c_3 is true again here this is A_1 , action is A_1 , but what if all are false then there is no action and so on.

We can complete this table and this is the decision table where the top part of the table contains the condition outcomes, the lower part this displays the corresponding actions. And each of the row is a rule and they form the test cases. So, as many rows will be there on our table, those many test cases we will need. So, this is a test case, this is a test case c_1 is false, c_2 is true and c_3 is true and so on. We are almost at the end at the end of this lecture.

We will stop here and continue our discussion from this point.

Thank you.