

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

**Lecture - 53**  
**Cause effect graphing**

Welcome to this lecture. In the last lecture, we discussed about the condition the decision table based testing, where we considered various combinations of conditions. We will proceed from there we will look at the Cause Effect Graphing. And, then we look at the pair wise testing, P way testing and pair wise testing, but before that let us do a small quiz based on the decision table based testing.

(Refer Slide Time: 00:52)

**Quiz: Design test Cases**

- Customers on a e-commerce site get following discount:
  - A member gets 10% discount for purchases lower than Rs. 2000, else 15% discount
  - Purchase using SBI card fetches 5% discount
  - If the purchase amount after all discounts exceeds Rs. 2000/- then shipping is free.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 144

So, the problem here is that we have a e-commerce site. And, here a member gets 10 percent discount on purchases lower than 2000 rupees. And if it is more than 2000 get 15 percent discount.

Purchase using some specific card like SBI card fetches additional 5 percent discount. And, if the purchase amount after all discounts exceeds 2000 then shipping is free, otherwise the shipping is charged. Now to develop the decision table based testing, we need to identify the conditions and the actions the next step would be to consider all possible combinations of conditions. And, then identify the actions corresponding to

those and we represent that on the decision table. And, each column in the decision table become becomes a test case.

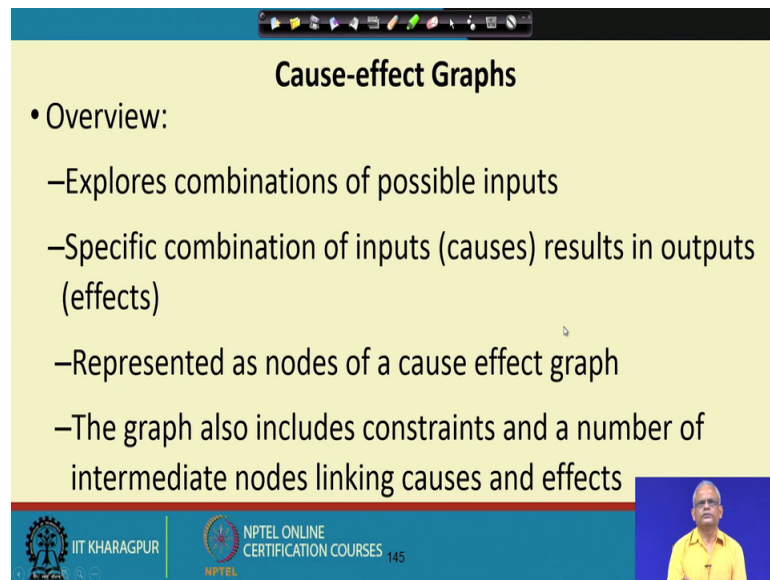
Now, let us first identify the conditions. If, we read here member gets 10 percent discount for purchases lower than 10,000. So, a condition here is that if the purchase is lower than 10000, 2000, then we will get if this is yes then 10 percent discount. If it is no then it is 15 percent discount, whether payment is made using SBI card this is another condition.

So, let me also write down the condition C 1 and C 2 and this fetches an additional 5 percent discount. And, the third condition is that the purchase amount after all discount exceeds 2000, this is the third condition. And, the action part is the decision is 10 percent, 15 percent, additional 5 percent and then the total amount exceeds 2000 and shipping is free.

We can write C 1 C 2 C 3 C 4 in the top rows of the decision table and A 1, A 2, A 3, A 4 are the actions. And, then if C 1 is true that is purchase is lower than 2000 C 1 is es C 2 is it is not purchased using SBI card, C 3 is the purchase amount exceeds after all discount 2000 rupees. We will say no and then we will see that the discount is 10 percent yes and the rest are no and so on.

We need to design this table, but just one word of question is that if this is no sorry if this is yes purchase amount is less than 10 percent. Then automatically the purchase amount of discount after discount will be less than 2000. So, we cannot have that this is yes and this is also yes. If this is yes this has to be no. So, there is a dependency between these 2 and that we have to take care while developing the decision table.

(Refer Slide Time: 05:03)



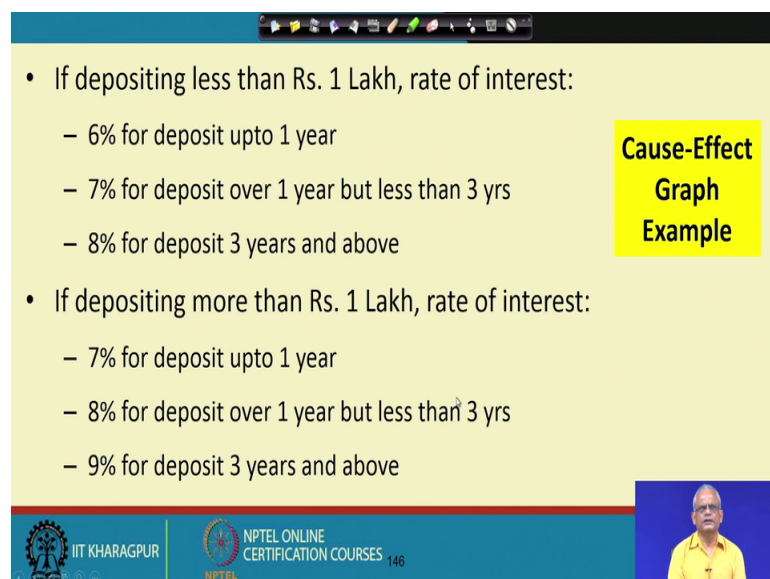
**Cause-effect Graphs**

- Overview:
  - Explores combinations of possible inputs
  - Specific combination of inputs (causes) results in outputs (effects)
  - Represented as nodes of a cause effect graph
  - The graph also includes constraints and a number of intermediate nodes linking causes and effects

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 145

As, we are mentioning earlier the cause effect graph provide a systematic technique and graphical symbol based on which we can explore the combinations of different inputs, represent them as nodes edges in a graph. And, then once the graph is developed we can easily translate the graph into a decision table. In the graph we can have constraints like and or etcetera and also we can have intermediate nodes between the input and output, which will help us to simplify the graph.

(Refer Slide Time: 05:53)



**Cause-Effect Graph Example**

- If depositing less than Rs. 1 Lakh, rate of interest:
  - 6% for deposit upto 1 year
  - 7% for deposit over 1 year but less than 3 yrs
  - 8% for deposit 3 years and above
- If depositing more than Rs. 1 Lakh, rate of interest:
  - 7% for deposit upto 1 year
  - 8% for deposit over 1 year but less than 3 yrs
  - 9% for deposit 3 years and above

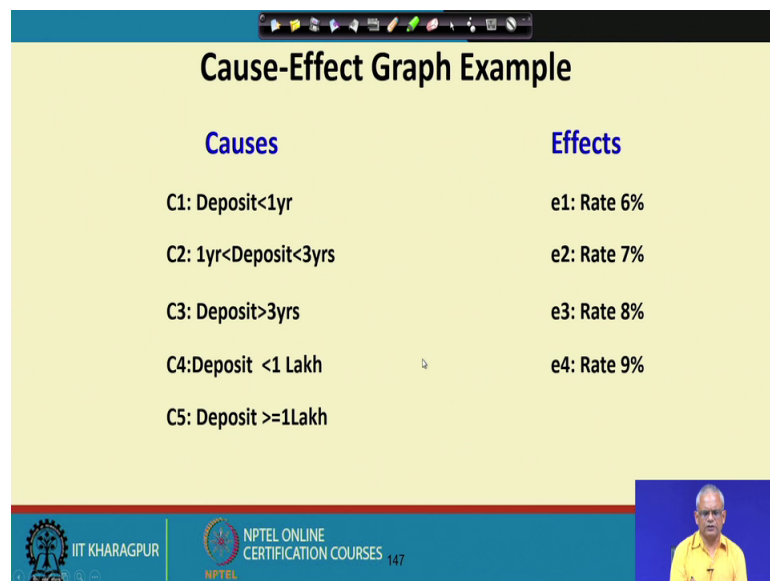
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 146

To explain this cause effect graphing using a simple example, that we were discussing earlier that if the deposit amount is less than 1000, then the rate of interest is 6 percent, 7 percent, 8 percent for deposit up to 1 year over 1 year and less than 3 year it is 7 percent 8 percent deposit is 3 years and above. But, if the amount is more than 1 lakh, then the rate of interest is higher 7 percent 8 percent and 9 percent.

Now, we want to develop the cause effect graph for this and then once we develop the cause effect graph we will translate that into a decision table. And, then the decision table will give us the test cases.

First let us do the cause effect graph, the cause is the conditions on the input and the action is the output. So, here the conditions and the input, if it is the amount of deposit is less than 1 year 1 to 3 year 3 year and also. Another condition is that, whether it is less than 1 lakh or more than 1 lakh.

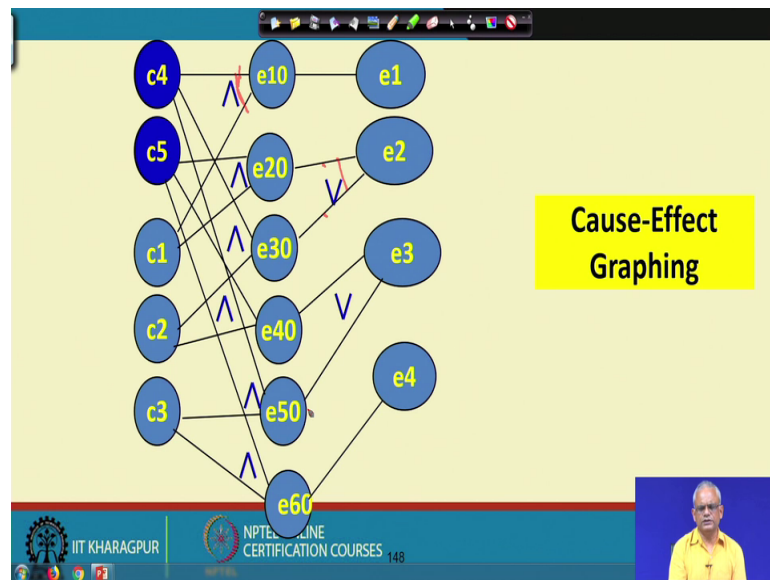
(Refer Slide Time: 07:16)



Causes	Effects
C1: Deposit<1yr	e1: Rate 6%
C2: 1yr<Deposit<3yrs	e2: Rate 7%
C3: Deposit>3yrs	e3: Rate 8%
C4:Deposit <1 Lakh	e4: Rate 9%
C5: Deposit >=1Lakh	

So, we represent that here the causes deposit is less than 1 year 1 to 3 year greater than 3 year and also, whether the deposit amount is less than 1 lakh or greater than equal to 1 lakh. And, we also note down the different effects or the actions, which are whether the rate applicable is 6 percent 7 percent 8 percent and 9 percent. Now, each of these we represent in from form of a graph.

(Refer Slide Time: 07:51)



We read the input conditions and the actions and then specific combinations of the input conditions; they give rise to specific actions.

For example, if it is less than 1 lakh and it is less than 1 year we write a and symbol here we write a and symbol here. So, that this is the constraint that both this conditions must hold. And, then the corresponding effect or the action is e 1. We have also a constraint in the form of or any of this if it holds then the action is e 2 and so on. Can easily translate the input into this graph form cause effect graph and once we have done the cause effect graph.

(Refer Slide Time: 08:58)

**Develop a Decision Table**

C1	C2	C3	C4	C5	e1	e2	e3	e4
1	0	0	1	0	1	0	0	0
1	0	0	0	1	0	1	0	0
0	1	0	1	0	0	1	0	0
0	1	0	0	1	1	0	1	0

- Convert each row to a test case

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 149

This can easily be translated into a decision table; we just identify from the graph that if certain causes are both 1, then the effect is 1 and so on. And, then once the decision table is stable out from cause effect graph, we get the test cases.

(Refer Slide Time: 09:20)

**Pair-wise Testing**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 150

Now, let us discuss how to reduce the number of test cases, because in some situations where the input data are too many. Input conditions are too many the number of test cases increases exponentially. And, that can become a very large number billions or

trillions of test cases and it becomes impossible to test the software. Even though we are able to design the test cases using a decision table based testing.

The pair wise testing helps us to reduce the number of a test cases.

(Refer Slide Time: 10:08)

**Combinatorial Testing of User Interface**

0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1	0
0	0	0	1	1	1	0	0	1	1
0	0	1	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	1	1	0	1	1

0 = effect off  
1 = effect on

**2<sup>10</sup> = 1,024 tests for all combinations**

\* 10<sup>3</sup> = 1024 \* 1000 ... Just too many to tests

Font dialog box showing 'Effects' section circled in red:

- Strikethrough
- Double strikethrough
- Superscript
- Subscript
- Shadow
- Outline
- Emboss
- Engrave
- Small caps
- All caps
- Hidden

NPTEL ONLINE CERTIFICATION COURSES 151

Let us see, how do we go about developing the pair wise test cases? In the motivation for pair wise test cases we just take the same example here that we are discussing earlier. That, we consider the font in a word processing software and the user has to choose, the specific font to be used, the style to be used, and the size of the font. And, also can use this checkboxes to indicate, whether it should be a strikethrough double strikethrough superscript subscript and so on.

Now, based on this how many test cases can be designed on a decision table based testing ok. We count here 4 3 7 3 10. So, there are 10 checkboxes here. And each of these either it is checked or unchecked. And, if we consider all possible combinations of this then it becomes 2 to the power 10. So, only based on the checkboxes it becomes 2 to the power 10 or 1 0 2 4, but then what about the other options? That also we must consider.

Let us assume that there are only 10 here it may be more, but for our simplicity. We just consider that there are only font, 10 font types to be selected and 10 styles to be selected and 10 font size to be selected and then, how many test cases? We need to multiply 2 to the power 10 with 10 to the power 3. And, this becomes 1 0 2 4000 too many test cases

for anybody to test the software. It will take years or 10s of years, but how do we go about testing this kind of software, because, these are very very common and I was saying that even controller software also have similar characteristic that many input variable combinations arise and different actions corresponding to those.

And, let see at how to reduce the number of test cases. Such, that it becomes possible to test this easily and also at the same time we do not want to lose the thoroughness of testing. We need a technique, which will reduce this 10 2 4000 into let us say just 8 or 10 test cases. And, there and also the thoroughness of testing should not suffer too much of course, it cannot be exactly as thorough as testing with all possible combination of conditions, but it should be 99 percent. Also, the thoroughness must be achieved and that is achieved by the pair wise testing, let us see the ideas behind this.

(Refer Slide Time: 13:44)

**Combinatorial Testing Problem**

$X_1$     $X_2$     $X_3$    ...    $X_n$

↓ ↓ ↓   ↓

**System S**

- Combinatorial testing problems generally follow a simple input-process-output model;
- The “state” of the system is not the focus of combinatorial testing.

IIT KHARAGPUR   NPTEL ONLINE CERTIFICATION COURSES 152

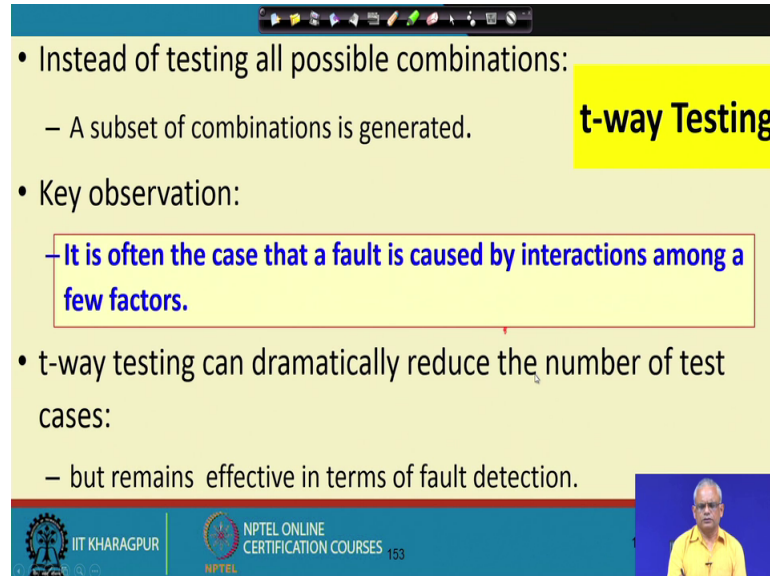
So, here the problem that we are addressing is that the system has only 1 state and there are many inputs and we need to consider the combinations on these input, but you may say that what if the system is state based; and there are multiple states of the system and transitions among state.

Then, the problem is more complicated we need to design the combinatorial test for every state of the system. And, also every transition and therefore, the problem becomes more complex, but we are now restricting to a simpler problem, where the system has



only 1 state and these are the inputs. And, we need to test using various possible combinations of the inputs.

(Refer Slide Time: 14:40)



The slide is titled "t-way Testing" in a yellow box. It contains the following content:

- Instead of testing all possible combinations:
  - A subset of combinations is generated.
- Key observation:
  - It is often the case that a fault is caused by interactions among a few factors.
- t-way testing can dramatically reduce the number of test cases:
  - but remains effective in terms of fault detection.

The slide footer includes the IIT KHARAGPUR logo, NPTEL ONLINE CERTIFICATION COURSES logo, and the number 153. A small video inset shows a man in a yellow shirt.

The key observation here is that often the bugs are caused by interaction among few factors. Even though we have 100s of input, but then some specific 2 combination among this might cause the bug or may be 3 combinations or in general t combinations may cause the bug.

And, if we consider all possible t combinations among the input, then we can dramatically reduce the number of test cases. And, still we will have a effective a bug detection, even very reduced number of test cases let us discuss this idea further.

(Refer Slide Time: 15:57)

The slide is titled "t-way Interaction Testing". At the top, a horizontal bar lists five parameters: "Interest Rate | Amount | Months | Down Pmt | Pmt Frequency". Below this bar, there are three colored boxes with text:

- A yellow box on the left: "All combinations: every value of every parameters".
- A green box in the center: "All pairs: every value of each pair of parameters".
- A purple box on the right: "t-way interactions: every value of every t-way combination of parameters".

Red lines connect the boxes to the parameter bar, illustrating the scope of each testing strategy. The text "etc. . ." is also present near the parameter bar. At the bottom left, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES 154. A small video inset of a speaker is visible in the bottom right corner.

Let us assume that a program or a function takes these as the input. Let us say it takes input interest rate, the amount borrowed the number of months for which the amount is borrowed and the down payment, and then the payment frequency. And, based on that it will let us say give us how much to be paid every month? That is the function.

If, we look at the parameters here there are 1 2 3 4 5 parameters. And, there are many possible equivalence classes boundary values and so on for each of this. If, we consider all possible combinations of conditions we represent this in a decision table and test, then it will become too many, but we can consider pair wise testing in which we have all possible combinations among 2 of input present. So, we have all possible combinations of these 2, these 2, these 2, these 2, and so on, all possible combinations between any 2 any 2 if they are present, then it is called as a 2 way testing.

We similarly can have 3 way testing. If, we have all possible combinations of any 3 let us say this, this and this, this, this and this, and so on. All possible combinations among the different parameters, then we have a 3 way testing, similarly we can have t way testing.

(Refer Slide Time: 18:19)

**Pairwise Testing**

Pressure	Temperature	Velocity	Acceleration	Air Density
A	T1	1	10	1.1
B	T2	2	0	2.1
	T3	3	20	3.1
		4	0	
		5		
		6		

Pressure | Temperature | Velocity | Acceleration | Air Density

A T1 1 10 1.1  
B T2 2 0 2.1  
T3 3 20 3.1  
4 0  
5  
6

Pressure Temperature  
A T1  
A T2  
A T3  
B T1  
B T2  
B T3

2

IIT KHARAGPUR  
NPTEL ONLINE CERTIFICATION COURSES

Now, let us understand the pair wise testing further. Let us for simplicity assume that there are 2 possible classes of input for pressure A and B, 3 possible input equivalence class input for temperature T 1 T 2 T 3. For velocity there are 6 velocity classes acceleration are 10 20 30 etcetera air density is some 1.1 2.1 3.1.

Now, let us say we want to consider pair wise testing. And, we need to have test case, which will have any combination between 2 of these input present. Let us say let us consider these 2 pressure and temperature. So, a test case must have pressure a temperature T 1 pressure a temperature T 2 A T 3 and B T 1 B T 2 T 3. So, that is all possible combinations of pressure and temperature and there are 6 entries here.

Similarly, we might have let us say temperature and velocity. So, we will have 18 entries here. It is not necessary that, they may be there on a different test case, but it may be that they are on the same test case, some of these are present let us say T 1 when this is T 1 this is let us say velocity is 1 velocity, when temperature is T 2 velocity is 2 and so on.

So, given the test case set of test cases, we should be able to identify given any 2 possible input values any 2 parameters, we should be able to find all possible combinations of conditions. Any 2 may temperature, air density, velocity acceleration, or velocity air density.

In our setup test cases we should be able to identify all possible combinations of conditions. This please remember that this is different from all possible combinations, where we have 0 0 0 0 0 0 1 etcetera, all possible combinations of conditions and that is too many. Here, it is possible to drastically reduce the number of test cases.

(Refer Slide Time: 21:15)

Number of inputs	Number of selected test data values	Number of combinations	Size of pairwise test set
7	2	128	8
13	3	1.6 x 10 <sup>6</sup>	15
40	3	1.2 x 10 <sup>19</sup>	21

Now, let us see what is the reduction that is possible? If the number of input is 7 and each is a Boolean, then if we do a decision table based testing. Then the number of rules or combinations of conditions is 2 to the power 7 is 128.

But, if we develop the pair wise tests, then we can do with 8. If the number of input is 13 and each can take 3 possible values, then the number of possible combinations or the number of the columns on the decision table becomes 3 to the power 13. And, this we can write 1.6 into 10 to the power 6, because this becomes 3 to the power 2 into 3 to the power 11, and that is becomes 9 into 3 to the power 11, and then we can find that it becomes ok. We, we can write here as 3 to the power 2 into 2 power 6 into 3. So, that is 3 to the power 12 into 3 and this is 9 to the power 6 into 3 and we can find that it becomes 1 point 6 into 10 to the power 6.

It's 1.6 million test cases, but then if you do pair wise testing it becomes just 15. Significant reduction can be achieved million test cases it is very difficult to test, it will take years to run a million test case, but just see here we can test with almost as much effectiveness using only 15 test cases.

Similarly, if the number of inputs is forty each 1 tests 3 only, 3 values then the number of test cases is  $1.2 \times 10^{19}$ . And, if there is a team testing these test cases in their lifetime they cannot complete testing, but if you consider pair wise testing then we can do with only 21 test cases which is a manageable number. So, the point that trying to convey through this table is that the reduction in the number of test cases is significant is dramatic in pair wise testing over combinatorial test case design like decision table.

(Refer Slide Time: 24:34)

**Fault-Model**

- **A t-way interaction fault:**
  - Triggered by a certain combination of t input values.
  - A simple fault is a 1-way fault
  - Pairwise fault is a t-way fault where  $t = 2$ .
- **In practice, a majority of software faults consist of simple and pairwise faults.**

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Now, to gain further understanding into the t way testing 2 way or pair wise testing etcetera. Let us look at some more concepts a simple fault is called as a 1-way fault. So, as long as a parameter has certain value this fault occurs, but if certain combination of 2 inputs. Let us say when the temperature becomes 100 only when the temperature is 100 and also at the same time the pressure is 50 only in this situation the error occurs. If, the pressure is 100 and temperature 60 and so on the error may not occur. So, this is the 2 way testing will detect this.

A 1 way testing of course, cannot detect this because, we will have hundred as long as it reaches 100 in some other test case it has reached 50, we would have a 1 way testing. It is a important observation that in a majority of software it is a experimental empirical observation, that in majority of software the fault consists of simple and pair wise faults several researches have conducted experiments. And they found that vast majority of the

faults are either simple faults, where as long as that value is taken that input value is taken then it fails.

Whereas a pair wise fault, when 2 specific combination of input is taken then the software fails.

(Refer Slide Time: 26:57)

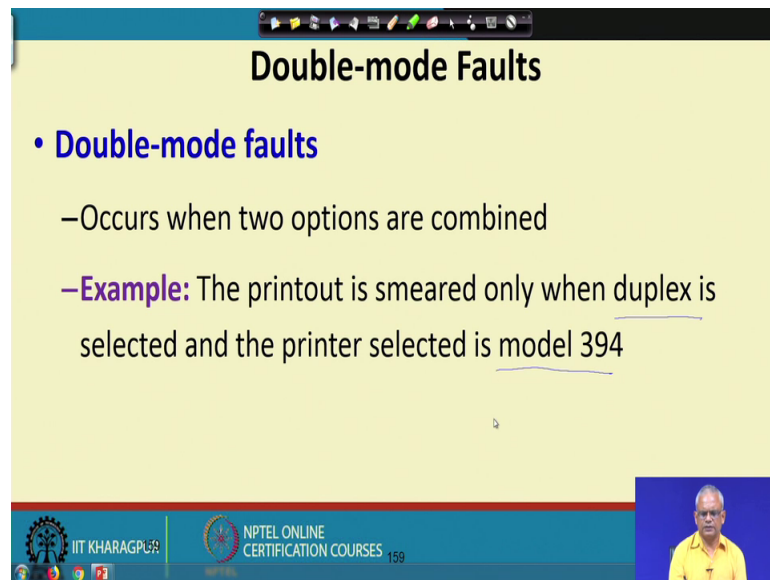
**Single-mode Bugs**

- The simplest bugs are single-mode faults:
  - Occur when one option causes a problem regardless of the other settings
  - Example:** A printout is always gets smeared when you choose the duplex option in the print dialog box
- Regardless of the printer or the other selected options

Let us see some example of a single mode bug. These kinds of bugs are very common here as long as this condition is satisfied, then the software fails one specific condition is satisfied the software fails.

For example, as long as the duplex mode is chosen in a printer irrespective of all other possible combinations, as long as it is a duplex option, then the printout gets smeared irrespective of all other options for example printer type color and so on. So, to test this we just need to check that every option is taken by the test cases the test cases give every option for all of the input.

(Refer Slide Time: 28:17)



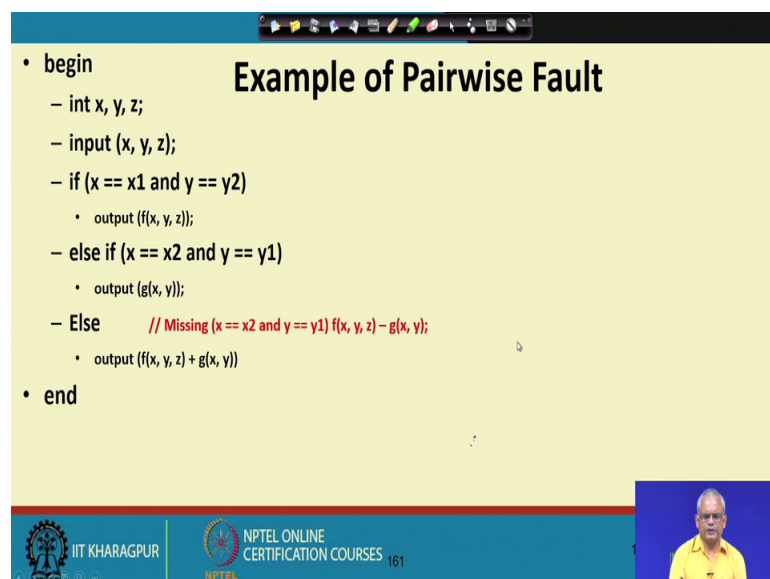
### Double-mode Faults

- **Double-mode faults**
  - Occurs when two options are combined
  - **Example:** The printout is smeared only when duplex is selected and the printer selected is model 394

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 159

A double mode fault here only if 2 specific combination of conditions are present then the problem occurs. Just to give an example the printout is smeared when the duplex is selected and the printer selected is 394. So, if you just select duplex and with some other model number problem does not occur few use simplex with 3 9 4 problem does not occur, but only for this specific combination these occurs. So, this can be detected by pair wise testing and in general.

(Refer Slide Time: 29:04)



### Example of Pairwise Fault

- begin
  - int x, y, z;
  - input (x, y, z);
  - if (x == x1 and y == y2)
    - output (f(x, y, z));
  - else if (x == x2 and y == y1)
    - output (g(x, y));
  - Else // Missing (x == x2 and y == y1) f(x, y, z) - g(x, y);
    - output (f(x, y, z) + g(x, y))
- end

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 161

And, we can have a  $n$  way testing. We will just look at that in the next lecture and we will look at an example of pair wise fault at the end of this lecture. And, we will stop here and continue in the next lecture.

Thank you.