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

Lecture - 09
Pair-wise Testing

Welcome to this session. So far we have been looking at black-box testing and we had looked at few black-box test strategies, but looked at equivalence class test cases. We have looked at special value test cases. We have looked at combinatorial test testing in the form of decision table based testing and cause effect graphing.

Now, let us try to look at one more black-box testing which is also a combinatorial testing applicable when the number of input is large is known as all Pair testing or Pair-wise testing.

(Refer Slide Time: 00:58)
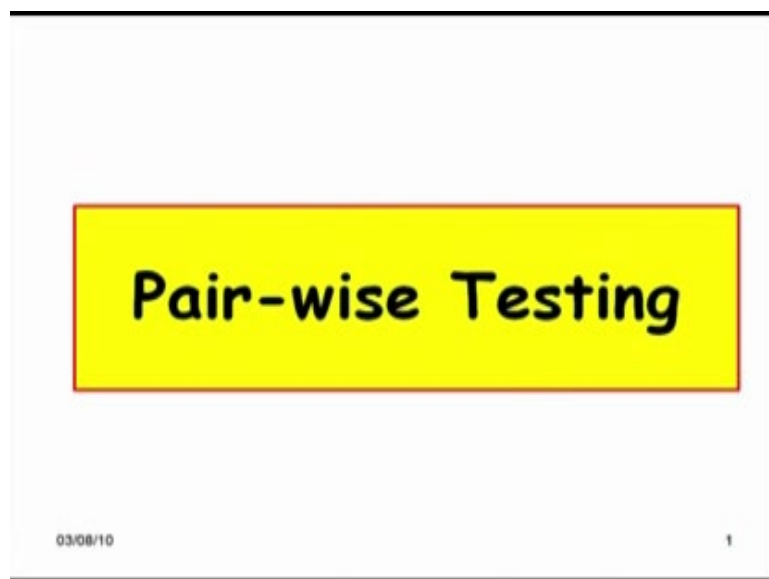
Let us look at the pair-wise testing strategy. As we are mentioning that many times we have situation, where we have many parameters and each parameter is either a Boolean or it can take several values. Let us look at this font setting in a word processor.
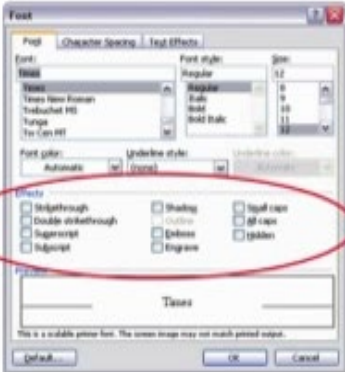
Each of these independently of the other we can switch on or off, strike through, double strike through superscript, subscript, small, caps and so on. So, why we can select 1 here, here and so on and not only that we can select any value here that is the font type, the fonts style you can select either regular, italic, bold or bold italic and the font size there are many sizes may be 30, 40 of them.

Now, if we consider all possible combinations that will be a huge number. Let us, for example, let us say that these are 4, 3, 7, 3, 10. So, 10 input and let us assume that independently we can check them on and off for simplicity even though we can check only some combinations sometimes either a superscript or subscript and so on. Let us for simplicity assume that we can check, we can give any possible combinations of the input parameters here.

The number of possible test cases required just for this much, these values is 2 to the power 10 or 1024, but with that we have to also consider the font colour or to consider the font itself, if we assume that there are 10 here. Let us assume that there are 10 here and here and here, let us assume that these 3 have only 10 values of course, there are more than here for simplicity let us assume 10 each. So, we have to consider 10 to the power 3. So, which is equal to 1000 and similarly we have combinations also corresponding to that here. So, just considering this and this and assuming conservatively there are 10 possible values here, it becomes 1024 into 1000 which is about a million values.

So, it just too many we cannot really test all possible combinations. We cannot really have testers running all these million possible combinations just for checking 1 screen. So, what is the alternative?
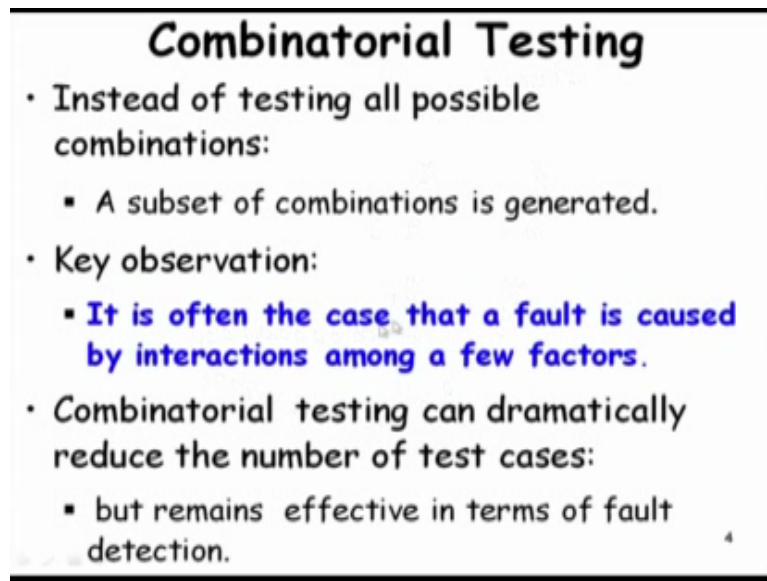
(Refer Slide Time: 04:34)
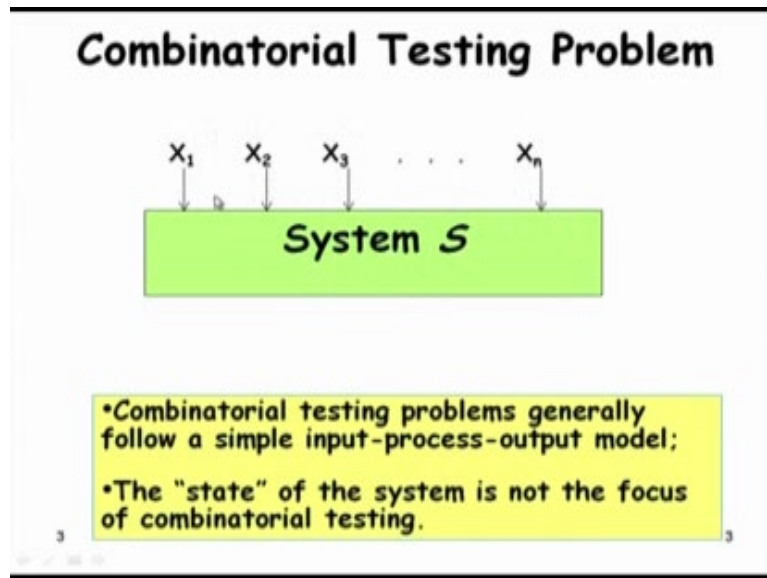
So, that is all about the pair-wise testing. To formally state the combinatorial testing problem that we have n inputs and each input can take some number of values, some valid values 3, some can take 2, some can take 5 possible values and so on and we assume that the system does not have state. The system S behaves the same way all the time when these possible combinations are given.

Otherwise the problem becomes much more complex which we can solve, but then first let us try to solve this simpler problem that the problem is state independent. The system that we are testing is state independent and then depending on the combinations of the values there can be bugs in the software.

Combinatorial Testing
- 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.
- Combinatorial testing can dramatically reduce the number of test cases:
  - but remains effective in terms of fault detection.

If there are n is large 20, 30 and so on and number of combinations are just too many to be able to test all possible values. So, how do we reduce the number of test cases so that we can do effective testing using 10, 12 or 20 test cases and still we able to recover, able to detect almost as much bugs as the million test cases. So, that is about the combinatorial testing and the key observation here is that when a program is written with the 30, 40 parameters it is some specific combinations of a pair of parameters or a single parameter or maybe up to 3 parameters that causes the problem. So, if 3 parameters, any 3 parameters that take ( ) specific value or any 2 parameters take some specific value or may be a single parameter takes a specific value then only the problem occurs.

All possible combinations we do not have to try out that researchers have experimented with large number of software and found that almost all bugs are found out if we consider 2 test, 2 combinations, if you have 40 input variables. So, let me just draw that.
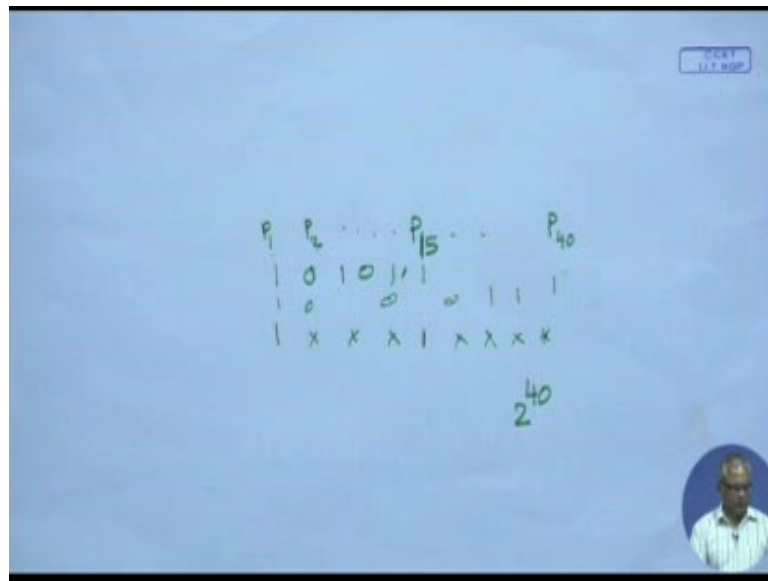
So, we have input parameter p 1, p 2 up to let say p 40 and each 1 takes either 0 or 1. Now, let us say a problem may occur if let say p 1 and p 15 they have value 1 1. So, for this possible combination p 1 and p 15, both getting set we have the problem. For any other combinations like this is 1, this is 1, this is 1, this 1 etcetera no problem occurs only ( ) when we have this specific setting, irrespective of the setting of the other inputs values whether they are 0 or 1 it really does not matter.

In that case to detect this kind of problem we need to really generate test cases which cover all pair-wise values like 1 15 set to 1 1, 1 15 set to 0 1, 1 15 set to 1 0 and p 1, p 2 set to 0 1 1 0 and so on. So, here with the number of test cases required will be much less because we might have test cases like 1 0 1 0 1 0 1 etcetera. So, this 1 test case itself as several pair-wise values 1 0 here 0 1 1 0 and so on. So, the number of test cases required to generate pair-wise combination of all possible values, 2 parameters will be drastically less.

So, for 40 parameters and each one taking 2 values we will need 2 to the power 40 test cases or considering all possible combinations of values, but if we consider only pair-wise values, all possible pair-wise values then the number of test cases may be 10 or 12 we do not know this a hard problem to know, but then we have tools available, we will just very simple small tools and run them and find out, they will give you the pair-wise test cases all pair test cases.

Let us look at the details of this. So, the assumption that only 2 or 3 variables when they have some value irrespective of the values of other variables we have the problem appearing that reduces the number of test cases and experimentally over large number of software it has been found that that is really the case that almost every bug gets detected when we consider combinations of 2 or at most 3. So, by 2 maybe large software might have got already 80 percent of the bugs, when you consider 3 combinations we might get 90 percent and by 4 or 5 we might have got all bugs. We do not have to if their 40 parameters do not have to consider all 40, experimentally it has been shown.

Let us look at the details of how do we what is involved in this pair-wise testing and how do we generate these pair-wise test cases manually and of course, they said that there are tools available where you just input the number of parameters and it will give you the test cases. So, the main idea why pair-wise testing works is that the fault is caused by interaction among a few factors. If you want to intuitively understand why it is so, if you look at the program code we do not really have the if case considering all possible combinations and then having cases for each of these, we have only if statements considering only few of the combinations of the conditions.

(Refer Slide Time: 12:16)
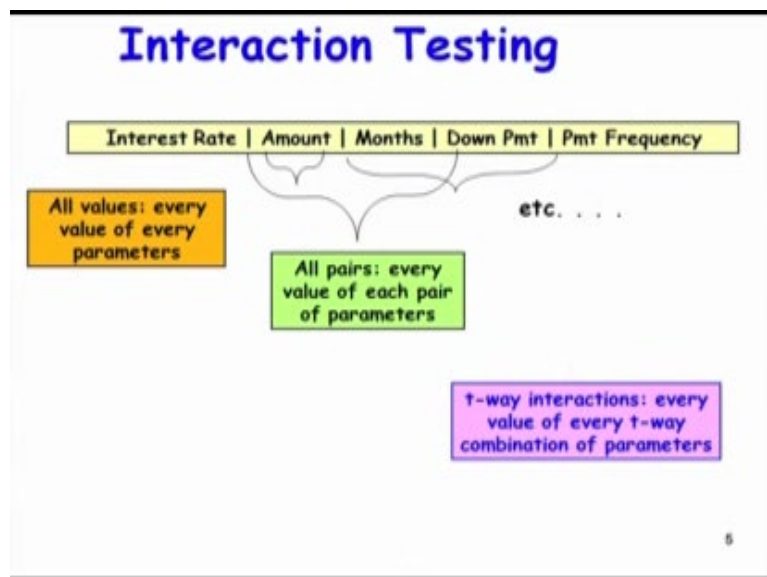
Let us consider this example that the interest rate, the amount, the months and then down payment and payment frequency. So, down payment is how much you pay at a time and what is the payment frequency, given this various parameters for a loan that what is the interest rate applicable becomes different. So, for the amount, the number of months what is the down payment and payment frequency? So, various combinations of these exist and we can consider pair-wise among these. Let

say interest rate and down payment, month and the payment frequency etcetera then we would have found out all the bugs.

(Refer Slide Time: 13:19)

Pairwise Reductions

Number of inputs	Number of selected test data values	Number of combinations	Size of pair wise test set
7	2	128	8
13	3	1.6×10^6	15
40	3	1.2×10^{19}	21

And the number of reduction and the number of test cases required is really dramatic. Just see here, if the number of inputs is 7 and each 1 can take 2 values these are Boolean, 7 Boolean inputs then the number of combinations is 2 to the power 7 which is 128, but the size of the pair-wise test set is 8. If you have 13 inputs and each test 3 the number of combinations is 3 to the power 13 which is 1.6 in 10 to the power 6, but then the size of the pair-wise test is 15 which is manageable just see million; 1.6 million and if it is 40 then and each 1 takes 3, just see here what is the number of test cases required, nobody in his lifetime can have such a system tested if each of these combinations have to be tested, but then the number of pair-wise test cases is 21, manageable somebody can run 21 test cases.

But then you might be wondering that how do I know that for 7 input and each one taking 2, I get 8 and for these 15, I get 21. How did I get these numbers actually this is generated by the tool and different tools might generate different number of test cases for this. So, finding how many pair-wise test cases will be required, a very hard problem, as I said that different tools can even generate different number of test cases depending on the algorithm they run, but then we can have algorithms that hill climbing combinatorial optimization like genetic algorithms and so on where we might get near optimal test cases.
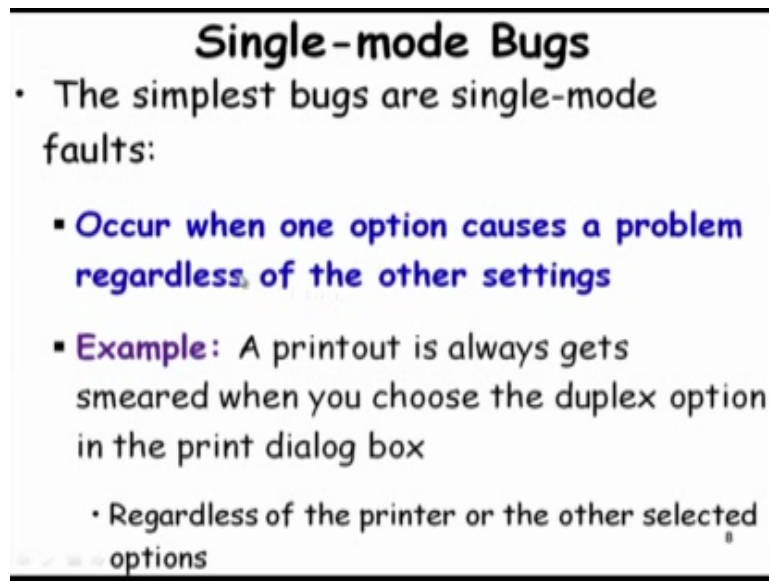
Fault-Model
- **A t-way interaction fault:**
  - Triggered by a certain combination of t input values.
  - A simple fault is a 1-way fault
  - Pairwise fault is a t-way fault where t = 2.
- **In practice, a majority of software faults consist of simple and pairwise faults.**

Now let us look at the nitty-gritty of this. So, we call it t-way interaction fault in a program if it is caused by some combinations of t of the input values. We might have n inputs values n may be 40 or 50 and out of that let say only 3 input values when they have some specific values 1 0 1, the problem occurs. The simplest t-way fault is a 1-way fault in a 1-way fault as long as 1 specific parameter is set to true or false or something then the problem occurs and a pair-wise fault is 2-way faults.

According to this terminology a t-way fault is caused by combination of t of the input parameters and as you are saying that, if we consider up to 5-way fault then all possible bugs would have been discovered in a practical software and as experiment shows that a majority of the software faults consist of simple and pair-wise faults, even for a very large software, complex and large software if we consider pair-wise faults we would have got 80-90 percent of the problem identified and if you consider 3-way or 4-way we would have got almost every bug that is present and at a very reduce number of test cases.

Now, let us look at an example of a single mode bug. So, a single mode bug occurs when one of the parameters is set to some value irrespective of the setting of all other parameters we have a problem.

Just look at an example that, the print out always gets smeared when you choose the duplex option in the printer dialogue box regardless of the printer type and other selected option. So, as long as you, one of the input check boxes you check that irrespective of the checking of all other, the problem occurs, and if you are not checked it you are not set this value duplex option then all possible combinations of other parameters will not cause any problem.

So, this is a single mode bug because it is caused by setting of a single parameter the problem appears a double mode fault or a 2-way fault occurs when two options are combined, for example, the print out gets smeared only when duplex is selected and the printer ( ) model is 394 only when 2 of this parameters have specific value the problem occurs or no other combination the problem occurs in a multi-mode fault 3 or settings, specific setting per 3 or mode parameters caused problem.

Now, let us looks at an example program to understand

(Refer Slide Time: 19:24)

Why this most of the problems are 2-way problems. So, this is a program which is trying to implement something and it is checking there are many parameters and it is considering only 3 of the parameters and then it is checking x 1, if x is equal to x 1 and y is equal to y 2 then it is f xyz. If it is x is equal to x 2 and y is equal to y 1 then the output is gxy. So, at this point the programmer has missed to write this statement else if x is equal x 2 and y is equal to y 2, I should have written here y 2, if x is equal to x 2 and y is equal to y 2 then the output should be f xyz minus gxy else output f xyz plus gxy because he has missed this x is equal to x 1 and y is equal to y 2.

So, the problem would occur. So, the programmer might either forget 1 of the, if class or maybe he would have written something wrong in the action part of the, if class. So, in that cases it would only when those specific settings are given it would cause the problem.

This is another example where Android smart phone has to be tested, the operating system has to be tested for various environmental settings. So, the operating system is configured by some environmental parameters like hard keyboard hidden, no to be set or hard keyboard hidden undefined to be set, screen layout is large or it is normal or it is small have to be set, or it is the orientation which is landscape or portrait. So, this is the configuration for android for a specific phone set.

Now, if we consider these variables which can take different values. So, we have 172,800 test cases. So, just too many, but you can try out the pair-wise testing, 3-way testing, 4-way testing, 5-way testing and by that we would have got almost every bug.

But then what we did not answer till now is that how are the pair-wise test cases generated, can we have some simple algorithm by which given a set of parameters and the values these parameters can take, can we get the pair-wise test cases. As I was saying that generating the optimum number of pair-wise test cases ( ) is a very, very hard problem and we can try out evolutionary algorithms, genetic algorithms and so on. But let us look at a simple algorithm which will give us most of the pair-wise test; I mean non optimal number of pair-wise test cases.

The first thing is to identify what are the variables, let us say we have orientation as 1 of the input parameter which can take 2 values portrait and landscape. We have screen as another parameter which can take 3 values and keyboard is another parameter which can take 2 values. So, the first thing we do is arrange this table, these are the input parameter table.

We arrange this table such that we first consider the parameter which takes the largest value if there are 2 which take largest will consider any 1 of them and then we arrange the next 1 and so on. So, we rearrange this table of input values such that the left most column has the largest number of possible values that it takes. So, here just an exhaustive testing will consider 2 into 3 into 2 numbers of test cases, but we can generate pair-wise test case which will take much less than that.

The first thing we did was we arranged the input parameters, the one that take maximum number of possible values that we arranged first, and one with next number of parameters and then the one with less or equal to this number of parameters, and remember that this is just a heuristic and we are just trying to generate manually test cases need not give the optimum number of test cases and also we have to manually check.

If the test cases are all right, otherwise you have to manually correct it and now in the next step we just arranged, we have taken the parameter which takes the maximum number of values and we have arranged it for the other parameter takes 2 values. So, I have written 2 of these here large, large, small, small, normal, normal and then this we have written alternatively portrait, landscape, portrait, landscape, portrait, landscape.

(Refer Slide Time: 25:59)

In the next step we take the third parameter and we just write down alternatively QWERTY, 12 key, QWERTY, 12 key, etcetera and then we can add more variables here and, but we have to make sure.

If all the pairs are present manually if not we make ( ) some adjustments here. So, that every pair of values between screen and keyboard screen and orientation, orientation and keyboard whether they exist are not. Otherwise we just adjust these values here little bit. So, that we get the pair of values or we can insert additional rows here to generate all possible combinations.

So, we have large and portrait and we have landscape and large and we have QWERTY and large we have 12 key and large and we have portrait and QWERTY, but what about landscape and QWERTY yes we have here. So, we just write down the values here and similarly, we do for the other parameters find out if any specific combinations are missing we try to add additional rules here.

(Refer Slide Time: 00:58)

So, this is a manually way of generating the test cases for all pairs testing. So, we can keep on adding pairs and then checking whether all possible combinations are existing or not.

(Refer Slide Time: 27:29)
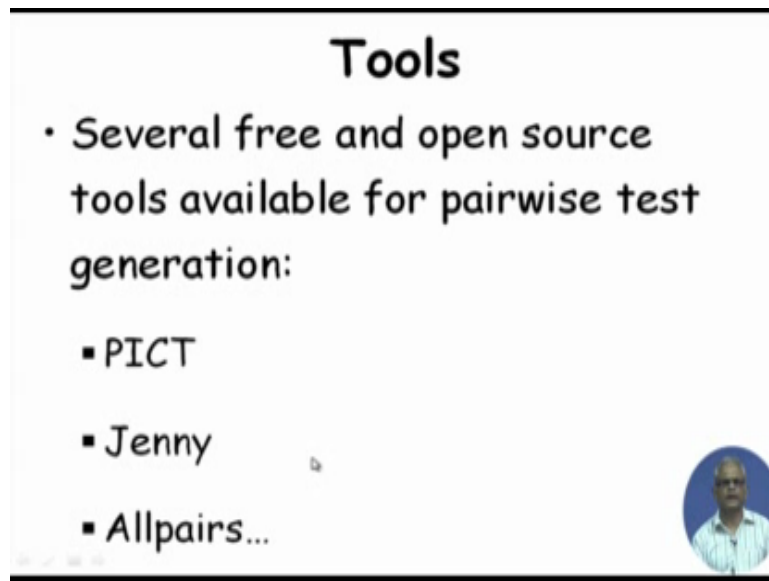
And finally, there may be duplicate test cases, we try to reduce the number little bit. So, this is a manually algorithm for generating test cases.

But then there are several small tools as they were saying they are available, you can just download them and run them and given the input parameters and the possible values. It will give you the exact test cases, the tools you can look on the Google PICT runs on the windows and I thing also on the Linux platforms, Jenny which is open source C program. You can see the Jenny program which as small C program possible 100-200 lines and you download the source code and run it and then all pairs is another tool.

There are several tools available, small tools and you can there will give you the number of test cases and as I was saying that you try with different values of input parameter and each parameter takes different number of values. Then these tools may give different number of test cases for them because they possibly use different algorithms.

But in any case, it will be worthwhile if you download this tool and run them it is quite easy small tool very usable tool. Most of these do not even have a GUI, that test interface. So, please do that and we will discuss about white box testing in the next session.

Thank you.