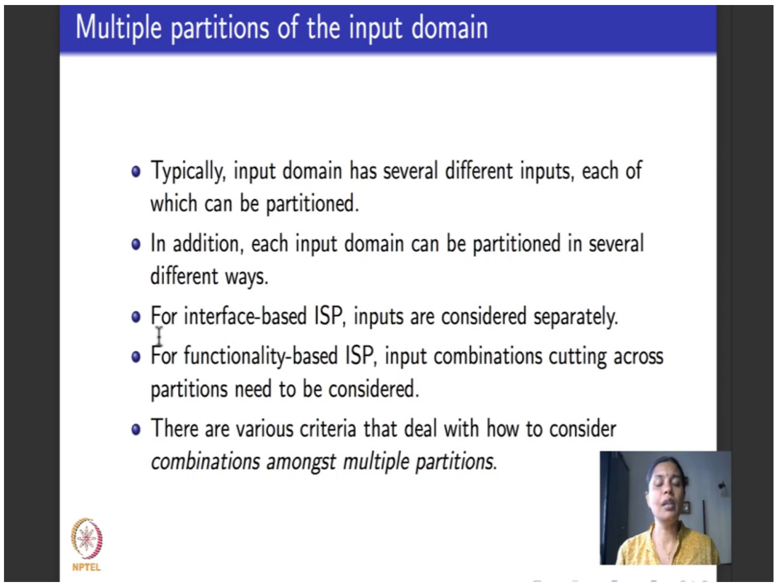
Software Testing
Prof. Meenakshi D’Souza
Department of Computer Science and Engineering
International Institute of Information Technology, Bangalore

Lecture - 33
Input Space Partitioning: Coverage Criteria

Hello again. Welcome to week 7, this is a fourth lecture. I will try and finish test case generation based on input space partitioning.

What did we look at till this week? We saw functional testing; I told you what are the various kinds of functional testing. Then I told you what is the basics of input space partitioning. In the last lecture we defined partitions. I told you how the input space could be partitioned in several different ways. One way we looked at was what was called interface domain part based partitioning; the other one was functionality based partitioning. So, we revisited the triangle type example. And then we saw how each of these work.

(Refer Slide Time: 00:52)



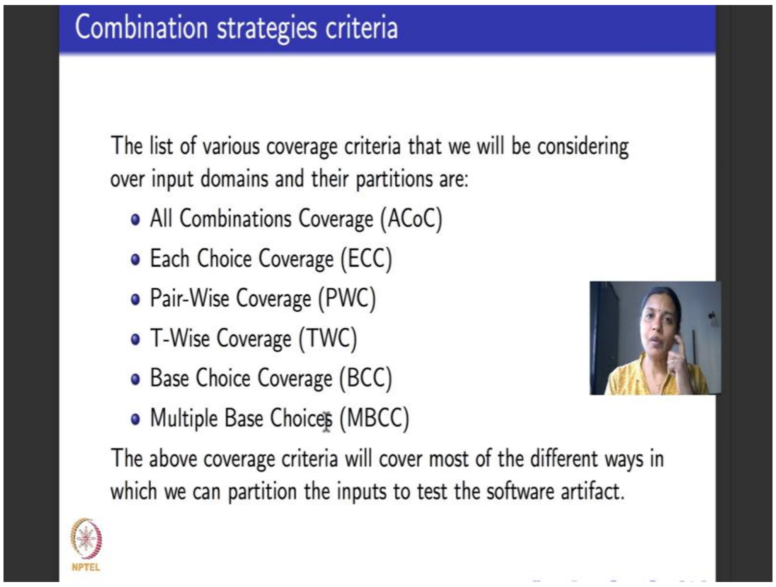
Multiple partitions of the input domain

- Typically, input domain has several different inputs, each of which can be partitioned.
- In addition, each input domain can be partitioned in several different ways.
- For interface-based ISP, inputs are considered separately.
- For functionality-based ISP, input combinations cutting across partitions need to be considered.
- There are various criteria that deal with how to consider *combinations amongst multiple partitions.*

Today what will be seeing is what are the various coverage criteria that we can define based on the partitions. Typically when I consider input domain there is one domain for every kind of variable that is a part of the input. And there are several different inputs each have their own types, each have their own domain and each of these input domains can be portioned in several different ways based on what is the requirement that I am considering and what is the requirement that I am trying verify.

For interface based input space partitioning we consider input separately as we saw in the last lecture. For functionality based input space partitioning we partition the input based on the concerned functionality of the program under test. So, now, what we will see is that how do we combine all these various partitions of inputs? Is there a clever way to combine, a better way to combine, an easier way to combine or a focused way to combine these so that we can test them.

(Refer Slide Time: 01:50)



Combination strategies criteria

The list of various coverage criteria that we will be considering over input domains and their partitions are:

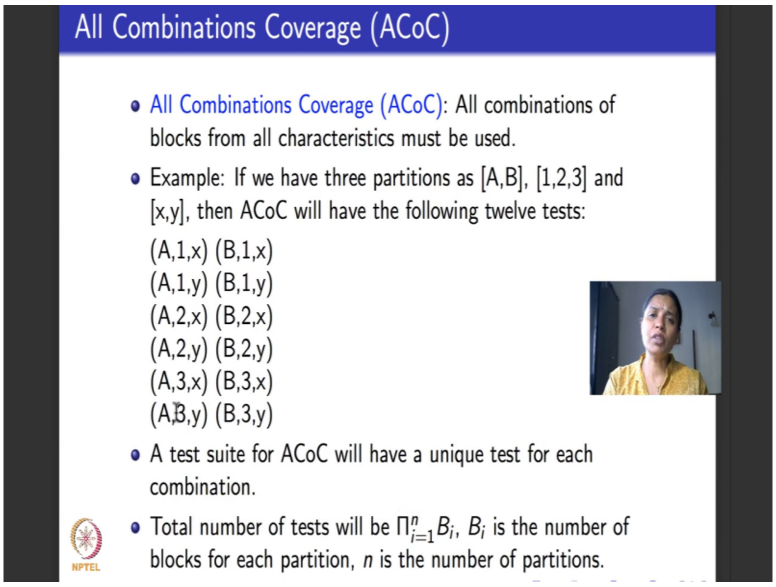
- All Combinations Coverage (ACoC)
- Each Choice Coverage (ECC)
- Pair-Wise Coverage (PWC)
- T-Wise Coverage (TWC)
- Base Choice Coverage (BCC)
- Multiple Base Choices (MBCC)

The above coverage criteria will cover most of the different ways in which we can partition the inputs to test the software artifact.

So, what we will be seeing today a various coverage criteria again define independent of any examples. I will explain these coverage criteria using an abstract example. In the next lecture we will take a concrete example which will be the triangle type program that we saw and see how these various coverage criteria can be used to define functional test cases for the triangle type program. So, the various coverage criteria that we will be defining in this lecture I listed here.

We will work with six coverage criteria. First one will be all combinations, second one will be each choice, third one and fourth one will take pairs or will take tuples of length t, the fourth and the fifth one. I mean the fifth and the sixth one will fix one coverage criteria like a base choice, and then define criteria based on that. Between these criteria almost all that pa ways of partitioning and input that we know off till date in test case generation will be covered.

(Refer Slide Time: 02:54)



All Combinations Coverage (ACoC)

- All Combinations Coverage (ACoC): All combinations of blocks from all characteristics must be used.
- Example: If we have three partitions as [A,B], [1,2,3] and [x,y], then ACoC will have the following twelve tests:
  (A,1,x) (B,1,x)
  (A,1,y) (B,1,y)
  (A,2,x) (B,2,x)
  (A,2,y) (B,2,y)
  (A,3,x) (B,3,x)
  (A,3,y) (B,3,y)
- A test suite for ACoC will have a unique test for each combination.
- Total number of tests will be $\Pi_{i=1}^{n} B_i$, B_i is the number of blocks for each partition, n is the number of partitions.

So, we begin with all combinations coverage, abbreviated as AcoC. What does is it do? It says you take input domain you partition the input domain the of the various inputs, you consider every combination from every partition. So, it is just this exhaustive testing based on partitions of the input domain. So, all combinations of all the blocks of partitions with respect to all the characteristics must be used to be able to test it for AcoC.

Suppose, let us take for some piece of software, for some input we have the following 3 partitions. Let us not worry about what the input is just an abstract representation of the 3 partitions. One partition partitions into 2 categories into sets called A and B, another input is partitioned into 3 sets call it 1, 2, 3, the third input is partitioned into 2 sets call it x and y right. So, there are 3 inputs: first input is partition in 2 ways A and B, second input is partitioned in 3 ways 1, 2 and 3, third input is partitioned in 2 ways again, x and y. What does all combinations say? It says take every partition with every other partition to test.

So, what I do for the first input I choose partition A and for the first input I choose partition B here, then I keep the partition for the second and the third fixed to be one and x, 1 and x. So, here I vary the partitions for the first input from A to B, similarly for the second input, I consider partition one, I fix partitions 1 and y for the second and third input I again vary the partition one for the first input from A to B. Similarly, what I do here? Now I vary the partitions for the second input I have made it changed it from one I have made it 2 here I again have made it two. So, I consider A and B with partition 2 and x.

So, between the middle column I cover the partitions of the second input, in the third column I cover the partitions of the third input, in the first tuples here that I am running my cursor down I cover the partitions of the first input. So, the first input has 2 partitions A and B, I have varied them thoroughly, second input has 3 partitions 1, 2 and 3 the second tuple in each of these test cases cover that, the third input domain has 2 partitions x and y, the third input in each of these tuples cover that.

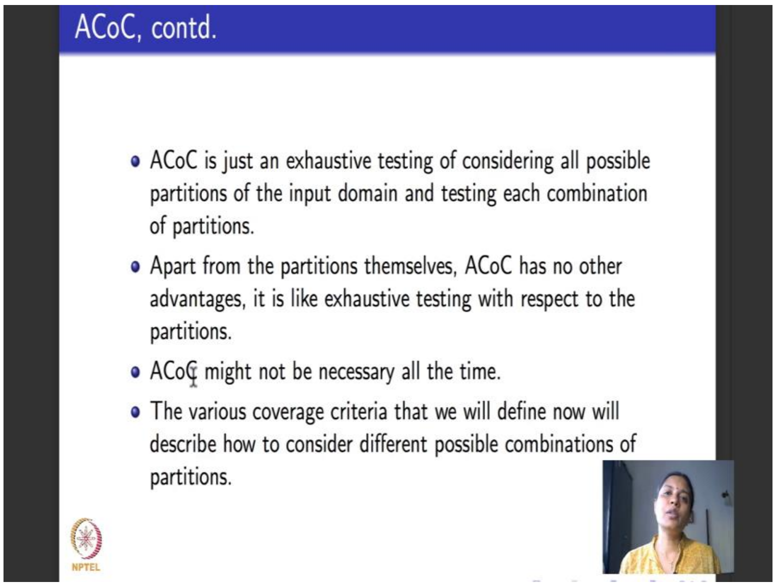
So, to be able to, this is a test requirement for all combinations coverage criteria. To be able to actually test it, what do I have to do? I have to pick up one test case for each of these petitions. So, I have to pick up one test case where the value is from, value for the first input is from A, here one test case whether value of the second input is from one and one test case whether value of the third input is from the partition x. Similarly for the second TR, I have to pick up one test case whether value of the first input is from partition B, the value of the second input if from partition one and value of the third input is from partition x and so on.

So, now I you see how many different partitions can be there for all combinations coverage criteria? Let us say each input is partitioned in B i ways, each input is partitioned in B i different ways and then there are n inputs, then the total number of partitions that I am targeting in all combinations coverage criteria will be a product of all these values B i. This ∏ that is written here means it is a product, it is a product

$\prod^{n}_{i=1} B_i$ where B_i is the number of blocks that are there for each partition.

So, if you go back to this example, the first input has 2 partitions number is 2, B i is 2 for that, second input the B i which is b 2 is 3 for that, for the third input the third partition B 3 is 2. So, the total number of test cases will be 2 into 3 into 2 that will be 12 and if you see 12 test cases have been written here.

(Refer Slide Time: 07:03)

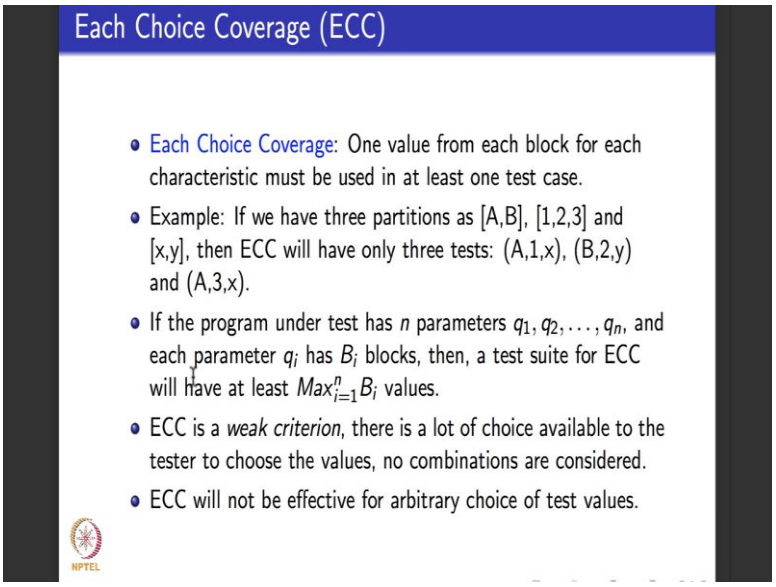

ACoC, contd.

- ACoC is just an exhaustive testing of considering all possible partitions of the input domain and testing each combination of partitions.
- Apart from the partitions themselves, ACoC has no other advantages, it is like exhaustive testing with respect to the partitions.
- ACoC might not be necessary all the time.
- The various coverage criteria that we will define now will describe how to consider different possible combinations of partitions.

Now, what is ACoC? ACoC or All Combinations coverage Criteria is not intelligent at all, it just tells you generate the partitions and just do an exhaustive testing with reference to all partitions. It is somewhat like doing all combinations coverage that we differ logic coverage criteria if you remember. Obviously, if we generating the partitions to be able to test them in a slightly intelligent way, we not generating the partitions to be able to test them before all possible combinations.

So, ACoC is not considered a great test requirement coverage criteria and it really has no advantages. We begin with it because we define, we would like to understand it for completeness sake and it is one of the value will coverage criteria. Whenever the partitions are small maybe we could afford to do ACoC coverage criteria. And the thing is it may not be necessary all the time. Now, how do we prevent ourselves from testing all the possible combinations? Are there better choices?

(Refer Slide Time: 08:01)



Each Choice Coverage (ECC)

- Each Choice Coverage: One value from each block for each characteristic must be used in at least one test case.
- Example: If we have three partitions as [A,B], [1,2,3] and [x,y], then ECC will have only three tests: (A,1,x), (B,2,y) and (A,3,x).
- If the program under test has n parameters $q_1, q_2, \ldots, q_n$, and each parameter q_i has B_i blocks, then, a test suite for ECC will have at least $Max_{i=1}^{n} B_i$ values.
- ECC is a *weak criterion*, there is a lot of choice available to the tester to choose the values, no combinations are considered.
- ECC will not be effective for arbitrary choice of test values.

Now, the next coverage criteria that we will see is what is called each choice coverage abbreviated as ECC. ACoC says you take all combinations of all the partitions, each choice coverage is the other extreme of ACoC. It says from each partition you pick up only one value. So, what is each choice coverage say, it says you pick up one value from each block, block is the same as partition, you pick a one value from each block for each characteristic and use it as your TR.

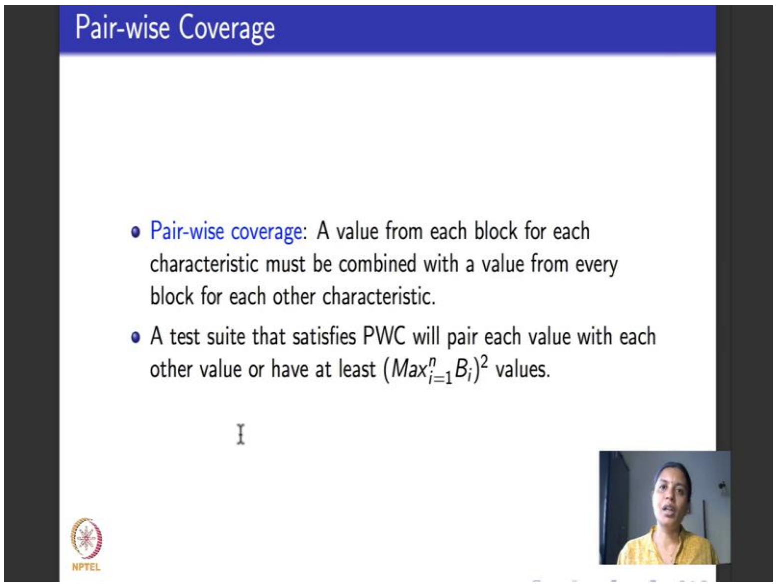
So, for the same example that we had that the first one was partition was A B, second one was 1, 2, 3, third one was x, y each choice coverage will have only 3 test cases. Randomly pick up, let us say I have picked up A for the from the first partition, I picked up 1 from the for second partition, and I picked up x from the third partition. In this case I pick up B from the first partition, 2 from the second partition let us say Y from the third partition.

Now, I have more or less covered the 2 partitions A and B and x and y. The only thing that I have not yet covered is this partition 3 for partitioning the second input. So, I pick up that for the third case and I can substitute either A or B for the first one, or x or y for the second one. So, I have chosen A, 3, x. So, each choice coverage says pick one value from each block. That is all we have done here. So, is the program under test has n parameters let say q 1 to q n, and each parameter has let say B i blocks or B i partitions then what will the test suite for ECC will have? It will have at least max of B i values.

Why is max? If you as you can clearly see here in the 3 partitions that we had for this example, only the second one 1, 2, 3 had cardinality three. So, to cater to that each choice thing we have to pick up one once 2 once and 3 once. So, and that is the maximum number. So, we need maximum of those test case values.

Like all combinations coverage criteria was like not a clever one, it included everything. ECC test the other approach it just says pick up anything randomly. So, it is actually a very weak coverage criteria. There is a lot of choice available on what you can pick up and what you need not and you might leave out some important combinations while testing with ECC. So, ECC will not be effective for arbitrary choice of test values.

(Refer Slide Time: 10:33)



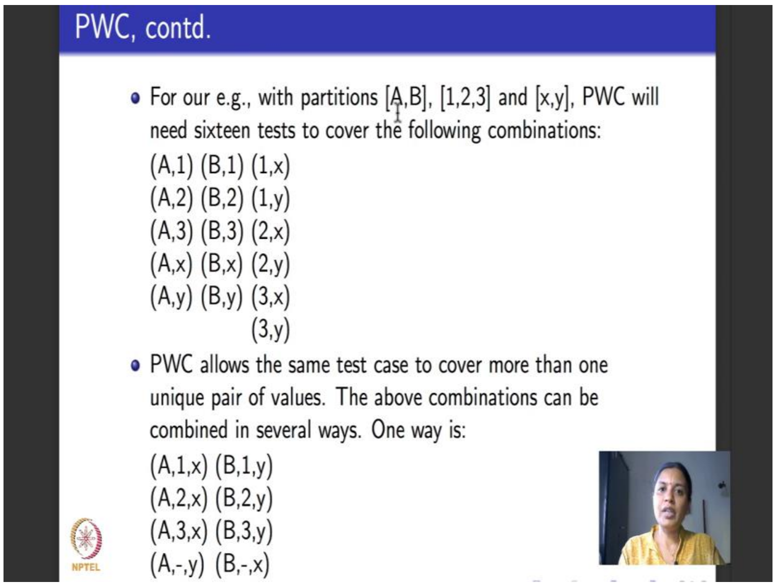
Pair-wise Coverage

- Pair-wise coverage: A value from each block for each characteristic must be combined with a value from every block for each other characteristic.
- A test suite that satisfies PWC will pair each value with each other value or have at least $(Max_{i=1}^{n} B_i)^2$ values.

So, what do people do? People look for midway options between ECC and all combinations coverage criteria. One midway option is what is called pairwise coverage. What happens in pairwise coverage? In pairwise coverage a value from each block of partition for each characteristic is combined with a value from every other block or partition for the other characteristic. So I take one pa block or partition corresponds to one characteristic of an input domain, I pick this value

Now, I list out what an all what are the other partitions that I have left out at. I park this value, fix this value and then pairwise combine it with all other values. So, a test suite that satisfies pairwise coverage, will have will pair each value with each other value or it will have how many test cases if B i are there, B i is the cardinality of each block or each partition, then it is clear that it will have max i is equal to 1 to n and B^2_i values right because I fix one B i and then I will let it vary over all the other partitions, then I fix the next one I will let it vary over all the partitions. So, it is B 1 into this B 2 into this and goes on. So, it will be max of B I, the whole squared.

(Refer Slide Time: 11:45)



PWC, contd.

- For our e.g., with partitions [A,B], [1,2,3] and [x,y], PWC will need sixteen tests to cover the following combinations:

  (A,1) (B,1) (1,x)
  (A,2) (B,2) (1,y)
  (A,3) (B,3) (2,x)
  (A,x) (B,x) (2,y)
  (A,y) (B,y) (3,x)
  (3,y)

- PWC allows the same test case to cover more than one unique pair of values. The above combinations can be combined in several ways. One way is:

  (A,1,x) (B,1,y)
  (A,2,x) (B,2,y)
  (A,3,x) (B,3,y)
  (A,-,y) (B,-,x)

So, for the same example we had this A,B, 1, 2, 3 and x, y, 3 different ways of partitioning pairwise coverage. How many test will I need? I will need 16 different tests. Why so? It the way they are arranged it will be clear to you what I have done here. I have fixed a to be the thing that I want to consider in the first partition, I let the second tuple vary I vary to over the second partition 1, 2 and 3, I vary it over the third partition x, y. In this column here I have fixed b is from the first partition, I vary it over the tuple 1, 2 and 3 and I have vary it over the tuple x, y for the third partition.

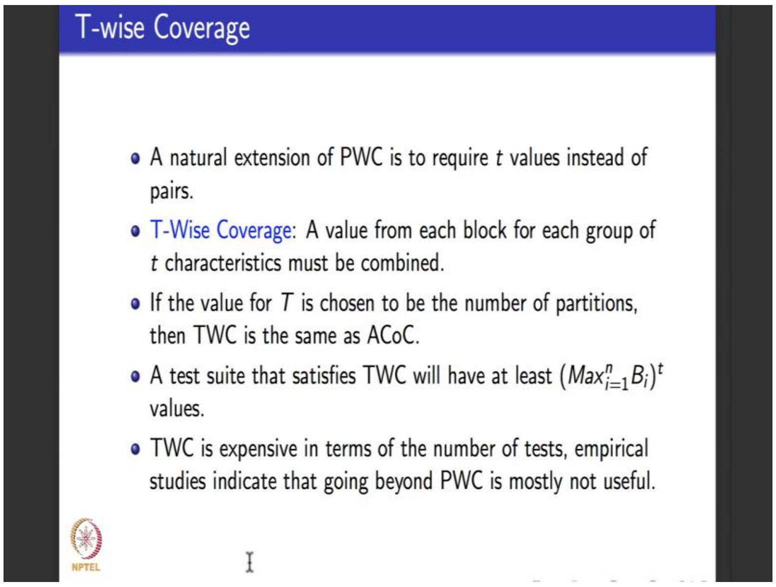
Now, between these I have not covered the way one varies with reference to the third partition x, y, the way 2 varies with reference to the third partition x, y, the way 3 varies with reference to the third partition x, y. That is what I have done here in the third column. In the third column you take combinations with reference to the partitions 1, 2, 3 and x, y and choose pairs that let each of them vary. So, here I fix one let x and y vary, here I fix 2, let x and y vary, here I fix 3 let x and y vary again, right.

So, if you count how many different test cases will be there, there will be so many different test cases. But remember one thing we are doing too much of work here. In fact, pairwise coverage what is let us do? It allows the same test case to cover more than one unique pair of values in some sense the above combinations that we have listed out here they can be combined in several ways. For example, I could directly do A, 1, x, in which case I have done paired A with 1, and 1 with x. If I do A, 2, x then I have paired A with 2 and I have paired 2 with x.

So, in the first case I do A with 1, 1 with x, I get A, 1, x, then I do A with 2, 2 with x, I get A, 2, x. Similarly A, 3, x considers both the pairs: a pairing of A with 3, and A pairing of 3 with x. So, once I have done here I just need to pair A with y, and because I have already paired A with 1, 2 and 3 once I have put a dash here in the last line, when I pair A with y. Read that dash as I can feel free to choose any of 1, 2 or 3 for that, because I have already paired A with each of them when I pair A with y, I can choose any one of 1, 2 or 3.

Similarly, in the other case what I have done here? We have paired B with 1 and y together, B with 2 and y, B with 3 and y. What is left to be paired? B and x; while pairing B and x because b is already paired with 1, 2 and 3, I put a dash to indicate that I can choose any of 1, 2 or 3 while pairing B, is that clear.

(Refer Slide Time: 14:43)



T-wise Coverage

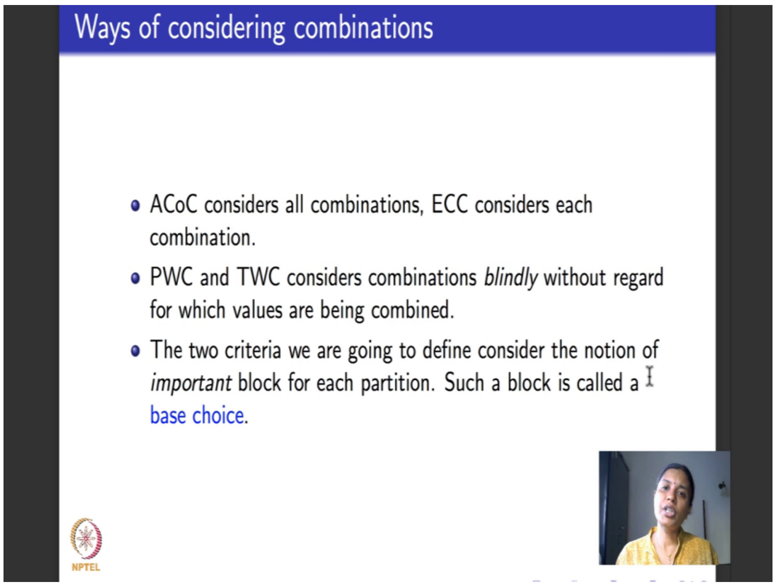
- A natural extension of PWC is to require t values instead of pairs.
- T-Wise Coverage: A value from each block for each group of t characteristics must be combined.
- If the value for T is chosen to be the number of partitions, then TWC is the same as ACoC.
- A test suite that satisfies TWC will have at least $(Max_{i=1}^{n} B_i)^t$ values.
- TWC is expensive in terms of the number of tests, empirical studies indicate that going beyond PWC is mostly not useful.

So, moving on pairwise considers only pairs or sets of 2. Whenever we to sets of 2 usually we can also extended to a arbitrarily large number. So, pairwise coverage can be extended to T wise coverage. So, instead of looking at a pair of values we require T values. So, what is T wise coverage? It says a value from each block or partition for each group of T characteristics must be combined. If the value for T is chosen to be the number of partitions that is the entire number, then T becomes all combinations coverage criteria.

If T is chosen to be one then T becomes each choice criteria. If T is chosen to be 2 then T becomes pairwise coverage criteria. A test suite that satisfies T wise coverage criteria as an extension of pairwise coverage criteria will have max over i is equal to one to n, B i to the power of T values where B i is the same, it is a cardinality for each of the partitions.

Again like all combinations coverage criteria T wise coverage criteria is considered to be an expensive coverage criteria, and is usually considered to be almost close to exhaustive testing or not necessary at all. So, people usually say that it is not wise to go beyond pairwise coverage criteria. An empirical study in software testing that involve input space partitioning also prove that T wise coverage criteria is not a very useful coverage criteria. But again we define it for the sake of completeness to indicate that whenever it is necessary it is possible to choose the T of your own. T could be 3, 4, 5 if we have lots of partitions and then do T wise coverage criteria.

(Refer Slide Time: 16:26)



Ways of considering combinations

- ACoC considers all combinations, ECC considers each combination.
- PWC and TWC considers combinations *blindly* without regard for which values are being combined.
- The two criteria we are going to define consider the notion of *important* block for each partition. Such a block is called a base choice.

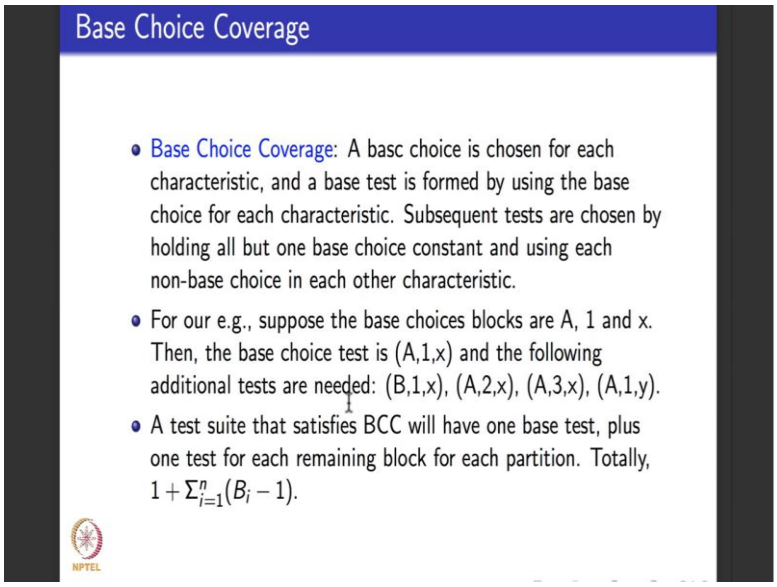
So, now what are the coverage criteria is that we have seen so far? We have seen four different coverage criteria. We began with all combinations coverage criteria which was like exhaustive testing with reference to combinations then we went on to the extreme down side. We said we will take each choice coverage criteria, pick up one choice from every partition, and then we did midway which was pairwise or T wise where the value of T can be chosen by the user.

Now, all these coverage criteria put together have a bit of a disadvantage. What is the disadvantage that they have? They consider the combinations blindly. What do I mean by blindly? They do not really think what is to be useful what categories of combinations would yield faults with a higher chance. Maybe you would want to focus on a particular set of input values. Can I focus and look at only those input values and consider the partitions with reference to the others? They do not look at all these they just look at things blindly.

Now, we look at 2 more coverage criteria that focus on avoiding these blind combinations of partitions. Those 2 coverage criteria will depend on fixing one partition as a base choice. So, the base choice is what we called as an important partition or an important block for a partition of a particular input and we define coverage criteria on that base choice. Now if you go back to our lessons on logic coverage criteria, base choice coverage counterpart in logic coverage criteria would be what is called active clause coverage. If you remember when we did logic coverage criteria, we said I want to focus on one particular clause and see how that clause influences the predicate and the clause that I want to focus on will be called active clause. Similarly we had inactive clause and we defined coverage criteria that will let each clause in turn to be reactive clause and test right.

Similarly, here when we look at partitions, they might be combinations or characteristics of partitions that I want to focus on to see how it influences particular pieces software under test. So, that block or partition that I want to focus on is called base choice coverage and we look at 2 different coverage criteria based on the base choice coverage.

(Refer Slide Time: 18:48)



Base Choice Coverage

- Base Choice Coverage: A basc choice is chosen for each characteristic, and a base test is formed by using the base choice for each characteristic. Subsequent tests are chosen by holding all but one base choice constant and using each non-base choice in each other characteristic.
- For our e.g., suppose the base choices blocks are A, 1 and x. Then, the base choice test is (A,1,x) and the following additional tests are needed: (B,1,x), (A,2,x), (A,3,x), (A,1,y).
- A test suite that satisfies BCC will have one base test, plus one test for each remaining block for each partition. Totally, $1 + \Sigma_{i=1}^{n}(B_i - 1)$.

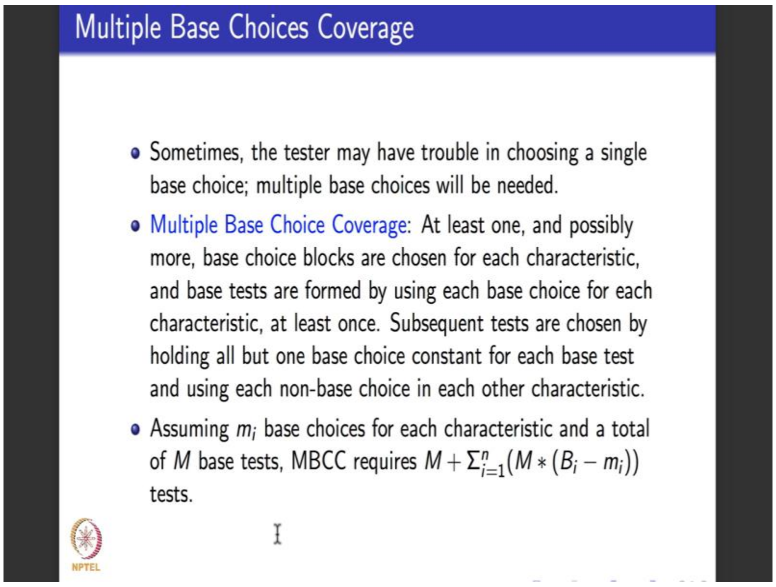
So, the first one is plainly called base choice coverage. What does it say? Say it is a base choice that is a partition of my choice is chosen for each characteristic, and a base test is formed by using the base choice for that characteristic.

Once you form the base test you keep it aside. Subsequent test which means all the other remaining tests how are chosen? They chosen by holding all, but one base choice constant and using each non base choice in each other characteristic. So, just to explain it in a different term, what do we do in base choice? The focus is I say I want to focus on one partition one characteristic call it the base choice right. So, for example, the base choice could be A, 1, x. Take that as a TR keep it aside, then subsequent test how do I choose the test that I want to add on to it? I choose by holding all, but one base choice constant right and choose each non base choice in the other characteristics. For example, if you consider the same partitions that we had which was a, b, 1, 2, 3 and x, y. Let us assume that my base choice is 1 A, 1 x this 3 tuple A, ,1 x. So, the base choice test is this.

Now, what do I do? Other test that I choose to be like this. So, what do I do? I fix one and x instead of a, I replace it with b right. In the second case I fix A and x instead of one I replace it with 2 and 3; and in the third case I fix A and 1 instead of x I replace it with y, is that clear? So, what I do is I choose one class of partitions call it the base choice keep it aside. Then what I do is I fix, I do not vary the other base choices vary one base choice. Like for example, once I have pick a one, x what are these remaining four tests? How do I obtain them after picking A, 1, x as my base choice. I park one and x, I vary A to B then in the second case I park A and x, I vary one to be 2 and 3 the other 2 blocks.

The third case I park A and 1 and I vary x to y. So, a test suite for base choice coverage how will I calculate how many tests are needed? If you see it will have one base test which like this which I have kept aside and then it will have one test for each remaining block of each partition that is what I have done here. So, totally the number of tests would be 1 + ∑ Bi – 1, where Bi is the number of blocks in each partition. Is that clear?

(Refer Slide Time: 21:36)



Multiple Base Choices Coverage

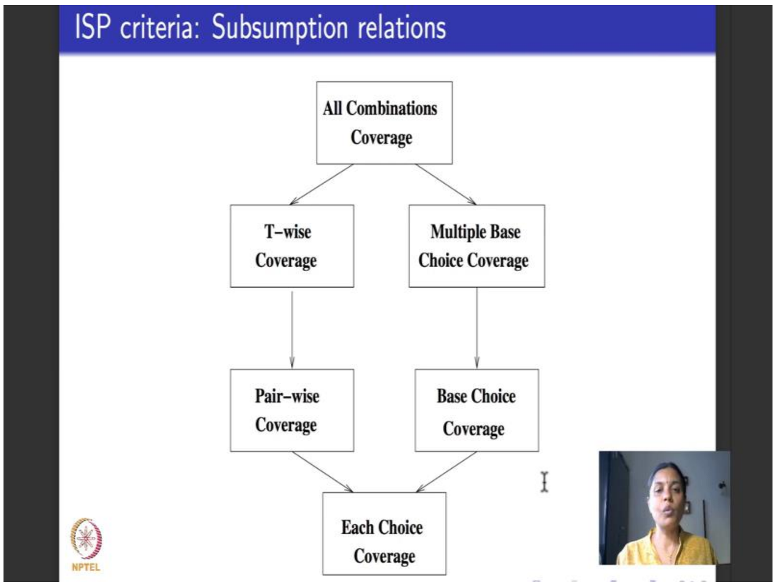
- Sometimes, the tester may have trouble in choosing a single base choice; multiple base choices will be needed.
- Multiple Base Choice Coverage: At least one, and possibly more, base choice blocks are chosen for each characteristic, and base tests are formed by using each base choice for each characteristic, at least once. Subsequent tests are chosen by holding all but one base choice constant for each base test and using each non-base choice in each other characteristic.
- Assuming m_i base choices for each characteristic and a total of M base tests, MBCC requires $M + \Sigma_{i=1}^{n}(M * (B_i - m_i))$ tests.

So, now moving on here we parked one base choice. There is nothing in that prevents from parking more than one partition as a base choice. If I park more than one partition of the base choice then I have what is called multiple base choices. So, what is multiple base choice do? Multiple base choices say that I have to pick up at least one, maybe more as base choice blocks for each characteristic. Then what do I do after that? Base tests are formed by using each base choice for each characteristic at least once.

So, that is like doing this A, 1, x where I had only one base choice. Here multiple base choices are there. So, you form you base test by using each of those multiple base choice in all the characteristics. After that what do I do? I do the same thing that I did for each base choice coverage. That is I hold all, but one base choice at any point in time constant and use each non base choice in each other characteristic.

For example, if you assume m i base choices for each characteristic and the total of capital M base tests, then multiple base choice coverage MBCC abbreviated as MBCC requires how many tests? This m is kept aside and then after that one I fix my M I can do B i minus M i and sum it to overall. Is that clear?

(Refer Slide Time: 22:57)


ISP criteria: Subsumption relations

So, this is how I do base choice and multiple base choice. So, what are the various coverage criteria that we have seen till now? We have seen all these six input space partitioning coverage criteria first one that we saw was all combinations coverage up here it was exhaustive testing with reference to partitions and all the characteristics. So, in terms of subsumption relation that subsumes all other coverage criteria. The next coverage criteria that we saw was each choice coverage criteria, which was the weakest of all the coverage criteria for input space partitioning because it just picks up one value from each of the partitions right and the choice of the values is completely random. So, it is quite a weak coverage criteria.

So, once I have pairwise coverage which means I park one value and let the others do vary and I consider them pairwise then I that extends each choice coverage criteria by definition and because T wise is any number assuming the T is any number greater than 2, T wise subsumes both pairwise and each choice coverage. In fact, as I told you T wise coverage will be equal to all combinations coverage if you consider T to be the cardinality of all combinations.

On the other side we do this base choice. It is like active clause coverage criteria on logic. I have fix one partition one characteristic as my base choice let the others vary. In plain base choice coverage, there is only one choice for the base in multiple base choice coverage there is more than one choice for the base. So, by definition multiple base choice subsumes single base choice I do not use the word single here, and there are no cross subsumptions here. Pairwise and base choice there is no relation, T wise and base choice there is no relation, similarly pairwise multiple base choice there is no relation and all combinations coverage definitely, because it is exhaustive testing subsumes multiple base choices and base choice coverage. Is that clear please?

So, what we will do in the next lecture is each of these coverage criteria I will take the triangle type example, and I will walk you through how the TR for each of these coverage criteria look like and how to write tests that satisfy test requirements for these coverage criteria.

Thank you.