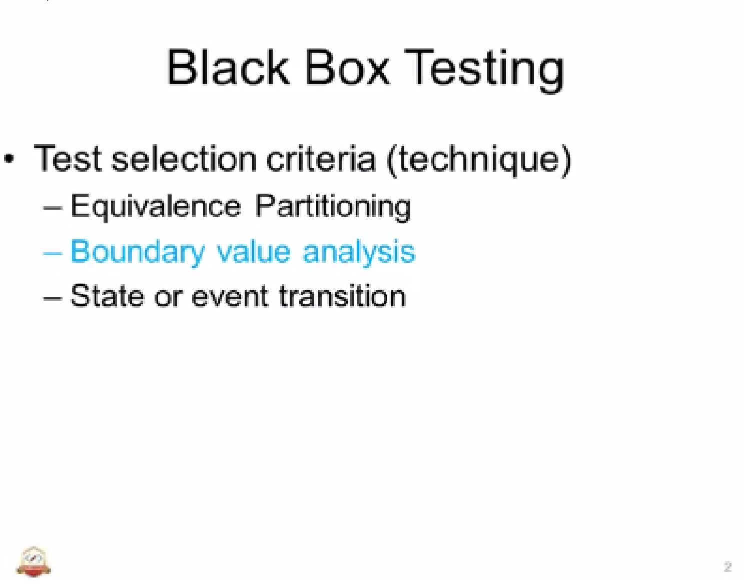
Embedded Software Testing
Unit 2: Testing Methods
Lecture 14
Seer Akademi-NPTEL MOU

Welcome you to the next session of embedded software testing unit 2 the testing methods lecture 14, this is the important unit because based on this the entire embedded software testing. A life is so we will be doing a detailed understanding and we played examples through off different testing methods. It is very important to understand the testing methods like what we spoke about black box testing and different techniques or refreshed techniques of black box testing.
(Refer Slide Time 00:47)

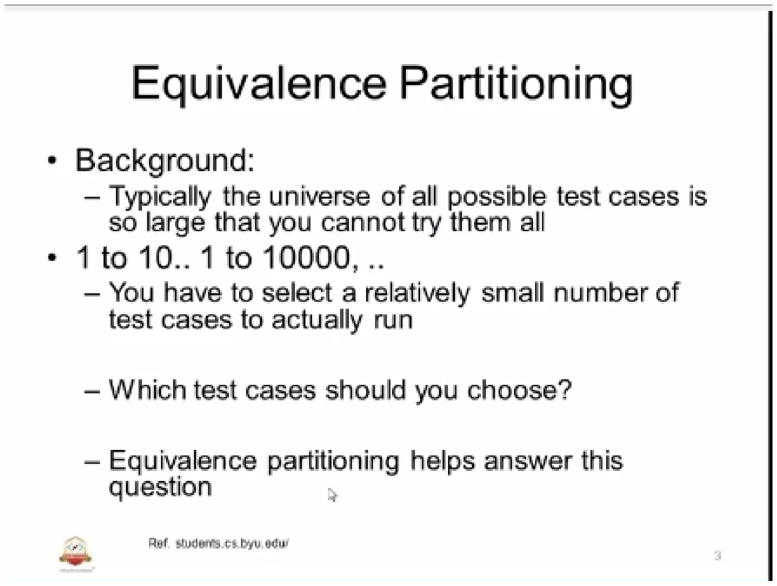


Black Box Testing

- Test selection criteria (technique)
  - Equivalence Partitioning
  - Boundary value analysis
  - State or event transition

2

Today we will study the boundary value analysis of this is in different technique, criteria technique again these are the platform is equivalence partitioning. What are the fundamental that we had understood in equivalence partitioning that includes accession being some of the values we consider differently. So before that we will just have a glance of what we studied in equivalence partition.
(Refer Slide Time 01:29)


Equivalence Partitioning

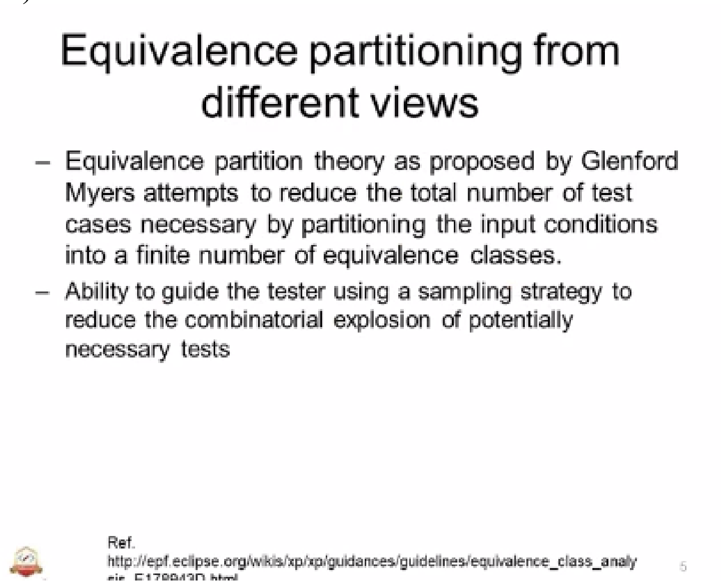
- Background:
  - Typically the universe of all possible test cases is so large that you cannot try them all
- 1 to 10.. 1 to 10000, ..
  - You have to select a relatively small number of test cases to actually run
  - Which test cases should you choose?
  - Equivalence partitioning helps answer this question

Ref. students.cs.byu.edu/

3

Equivalence partition we know why we get because we cannot afford to have numeric test cases simply because we have possibility of doing test to the different number of inputs because the system can behave or either the normal way or an abnormal way for a typical requirement. We do test only the sufficient levels of inputs how we can do that reduction is by having the partitioning that is called equivalence portioning.

(Refer Slide Time 02:12)


Equivalence partitioning from different views

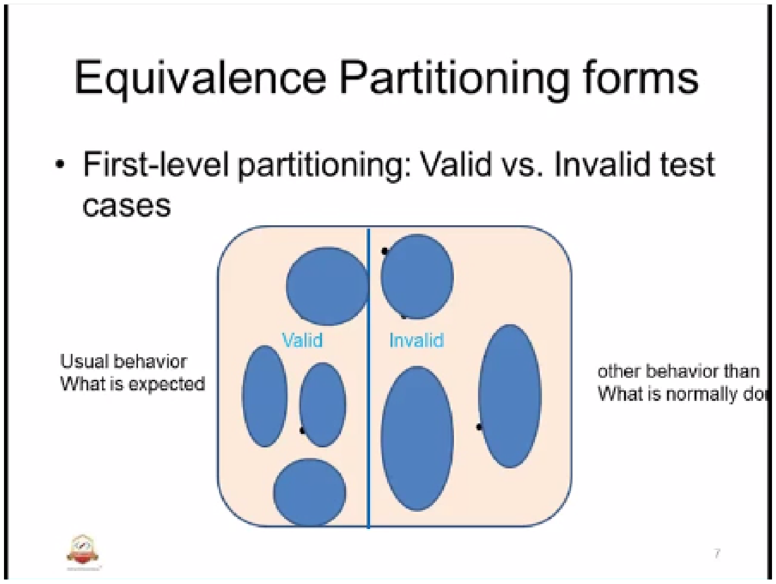
- Equivalence partition theory as proposed by Glenford Myers attempts to reduce the total number of test cases necessary by partitioning the input conditions into a finite number of equivalence classes.
- Ability to guide the tester using a sampling strategy to reduce the combinatorial explosion of potentially necessary tests

Ref. http://epf.eclipse.org/wikis/xp/xp/guidances/guidelines/equivalence_class_analysis_E1789430.html

5

Also we understood that the basic purpose is to reduce the total number of test cases by portioning the input conditions into finite number of equivalent classes.

(Refer Slide Time 02:24)

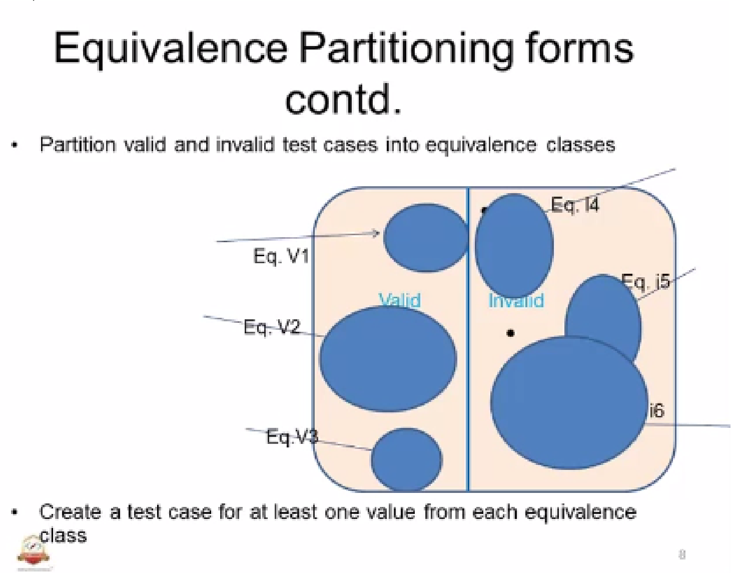


Equivalence Partitioning forms

- First-level partitioning: Valid vs. Invalid test cases

So we also studied about the first level of valid and invalid test cases first we are going to define all the test cases and then we are going to have a partition of one usual behavior what is expected for testing those inputs. Those are called valid equivalence partitioning the other one which is other than the normal behavior in term of outside the count are whatever it is those are called equivalence partitioning.

(Refer Slide Time 03:03)



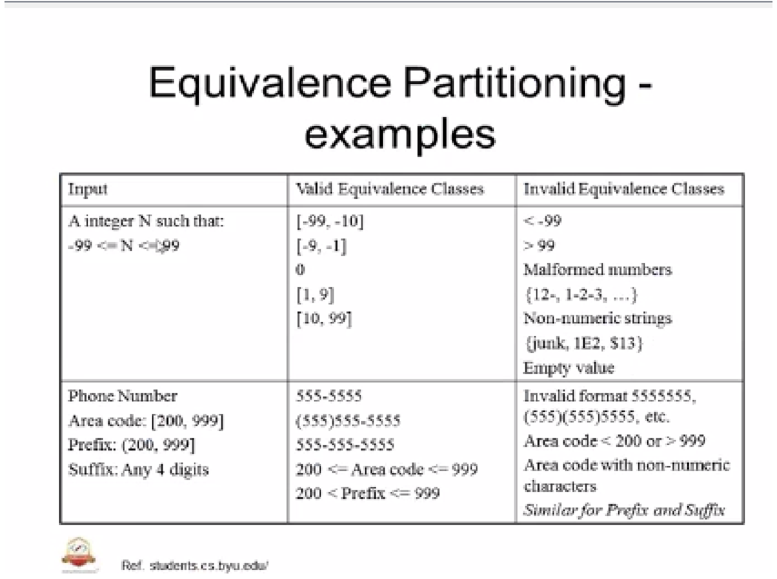
Equivalence Partitioning forms contd.

- Partition valid and invalid test cases into equivalence classes

- Create a test case for at least one value from each equivalence class

And we defined the numbers accordingly equivalent valid one, valid two, valid three, based on the classes and each classes will have one test case selected from

(Refer Slide Time 03:18)

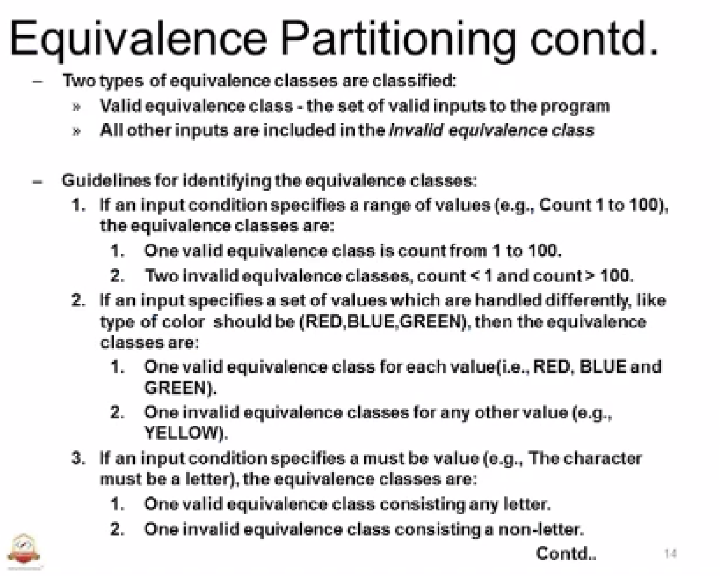


Equivalence Partitioning - examples

Input	Valid Equivalence Classes	Invalid Equivalence Classes
A integer N such that: -99 <= N <= 99	[-99, -10] [-9, -1] 0 [1, 9] [10, 99]	< -99 > 99 Malformed numbers {12-, 1-2-3, …} Non-numeric strings {junk, 1E2, $13} Empty value
Phone Number Area code: [200, 999] Prefix: (200, 999] Suffix: Any 4 digits	555-5555 (555)555-5555 555-555-5555 200 <= Area code <= 999 200 < Prefix <= 999	Invalid format 5555555, (555)(555)5555, etc. Area code < 200 or > 999 Area code with non-numeric characters *Similar for Prefix and Suffix*

Ref. students.cs.byu.edu/

And we also studied from example, of integer N, the engine from minus 99 to plus 99, so valid equivalent classes. So like this we have equivalent classes, similarly we have about three to four equivalents classes where we try the input the values such a way that it is testing, it is tested with an invalid arrange of inputs.
(Refer Slide Time 03:58)

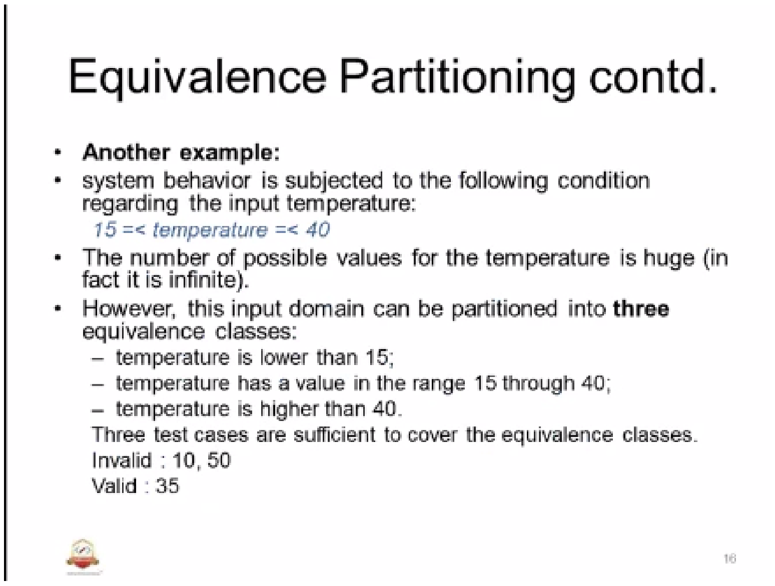


Equivalence Partitioning contd.

- Two types of equivalence classes are classified:
  - » Valid equivalence class - the set of valid inputs to the program
  - » All other inputs are included in the *Invalid equivalence class*
- Guidelines for identifying the equivalence classes:
  1. If an input condition specifies a range of values (e.g., Count 1 to 100), the equivalence classes are:
     1. One valid equivalence class is count from 1 to 100.
     2. Two invalid equivalence classes, count < 1 and count > 100.
  2. If an input specifies a set of values which are handled differently, like type of color should be (RED,BLUE,GREEN), then the equivalence classes are:
     1. One valid equivalence class for each value(i.e., RED, BLUE and GREEN).
     2. One invalid equivalence classes for any other value (e.g., YELLOW).
  3. If an input condition specifies a must be value (e.g., The character must be a letter), the equivalence classes are:
     1. One valid equivalence class consisting any letter.
     2. One invalid equivalence class consisting a non-letter.

Contd.. 14

Similarly other example of phone numbers we had studied and we also understood about guide lines there five guide lines that are important in terms of time in memory, sizes, range, count, etc…
(Refer Slide Time 04:13)



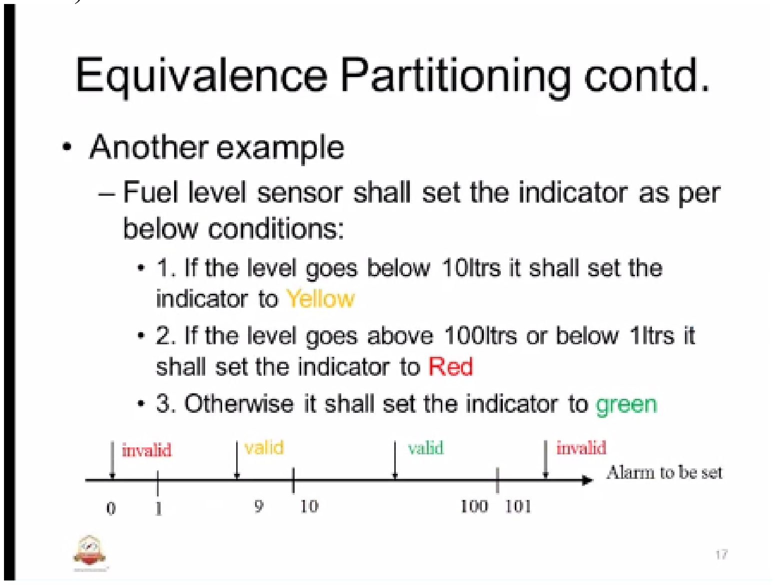
Equivalence Partitioning contd.

- **Another example:**
- system behavior is subjected to the following condition regarding the input temperature:
  15 =< temperature =< 40
- The number of possible values for the temperature is huge (in fact it is infinite).
- However, this input domain can be partitioned into **three** equivalence classes:
  - temperature is lower than 15;
  - temperature has a value in the range 15 through 40;
  - temperature is higher than 40.

  Three test cases are sufficient to cover the equivalence classes.
  Invalid : 10, 50
  Valid : 35

16

We took few more examples in terms of temperature from 15 to 40,
(Refer Slide Time 04:18)



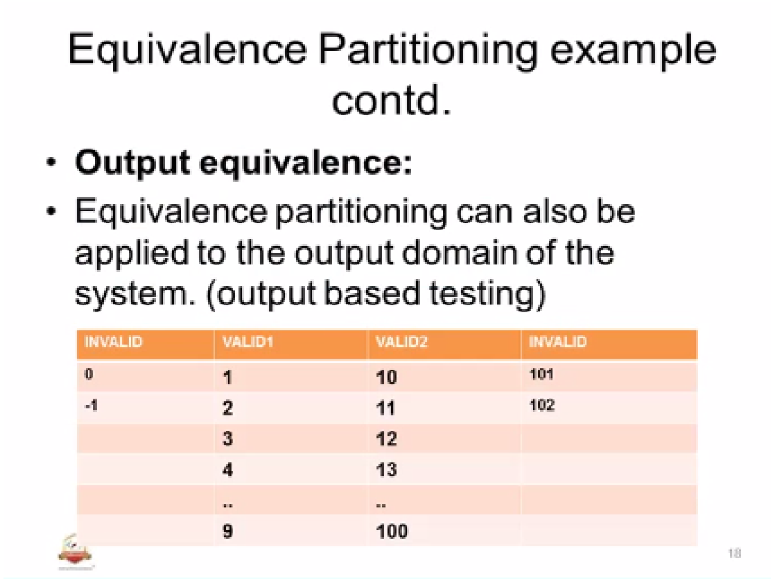
Equivalence Partitioning contd.

- Another example
  - Fuel level sensor shall set the indicator as per below conditions:
    - 1. If the level goes below 10ltrs it shall set the indicator to Yellow
    - 2. If the level goes above 100ltrs or below 1ltrs it shall set the indicator to Red
    - 3. Otherwise it shall set the indicator to green

invalid
valid
valid
invalid
Alarm to be set
0
1
9
10
100
101

17

Then we have fuel level sensors having three types of indicators yellow, red, and green. So we know that what is invalid? What is valid?
(Refer Slide Time 04:42)



Equivalence Partitioning example contd.

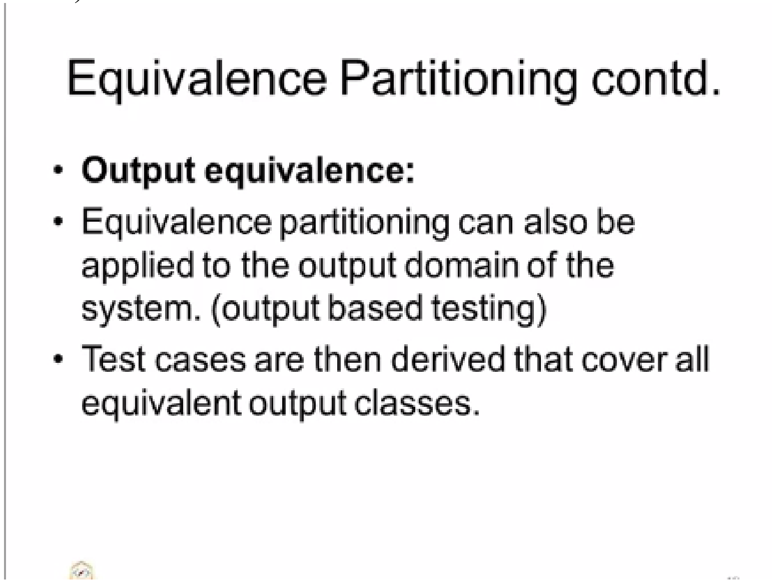
- **Output equivalence:**
- Equivalence partitioning can also be applied to the output domain of the system. (output based testing)

INVALID	VALID1	VALID2	INVALID
0	1	10	101
-1	2	11	102
	3	12	
	4	13	
	..	..	
	9	100	

18

So basically we are going to draw a range of all the test cases and again in the range we are going to create a table, it is called the trip table having all the possible completions defined for each of the inputs with the help of different columns. In further we are going to have the equivalence classes divided as valid, invalid test of the below table.

(Refer Slide Time 05:01)

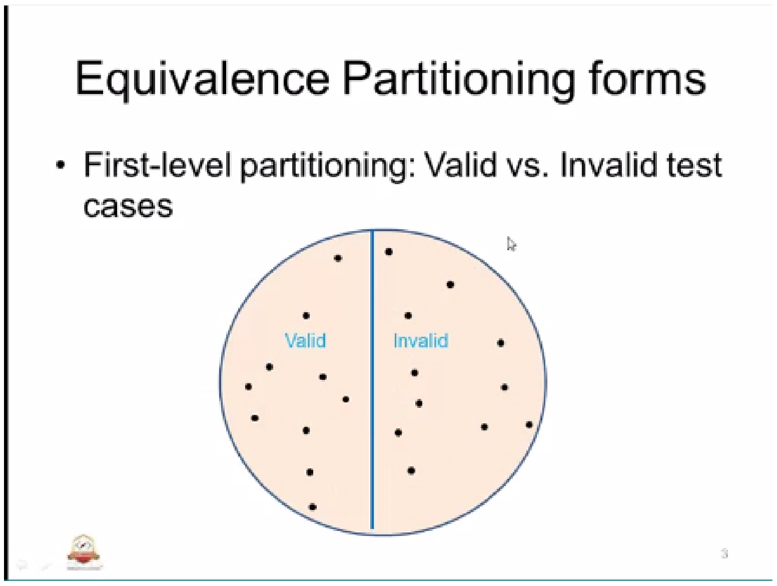


Equivalence Partitioning contd.

- **Output equivalence:**
- Equivalence partitioning can also be applied to the output domain of the system. (output based testing)
- Test cases are then derived that cover all equivalent output classes.

We also know that we can have the output equivalents classes defined as well not only the input dependence, so that is about equivalence partition.

(Refer Slide Time 05:29)

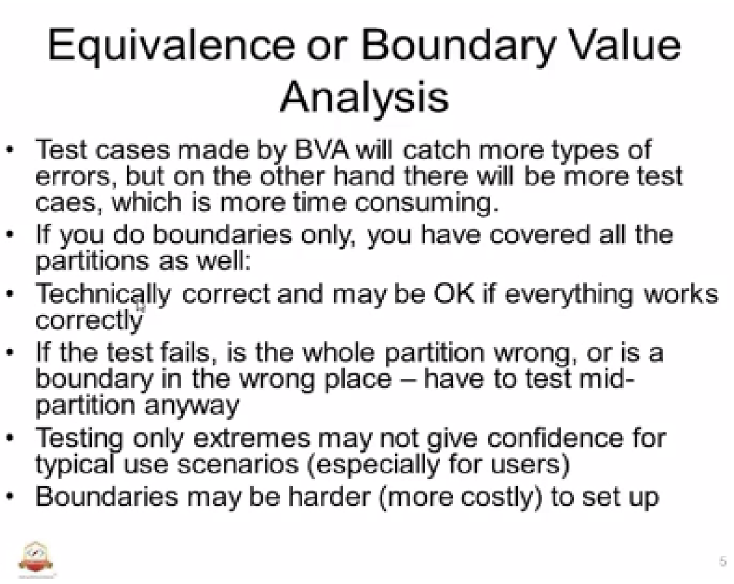


Equivalence Partitioning forms

- First-level partitioning: Valid vs. Invalid test cases

Valid
Invalid

Today we will go through the black box testing technique called boundary value analysis. So what is the boundary value analysis? First we understood that the equivalence partitioning forms with the help of valid and invalid functions.
Here the valid will have the numeric numbers of test cases you can see that black dots within the first five, and the second five as invalid test cases and the further we are going to group them in terms of, say this is one set, and this is another set, this is another set, so like this we have, say for valid some four are there for invalid there are equivalents valid one, and we have equivalents valid two and equivalents valid three, etc…
So we have equivalents invalid data, one we have equivalents invalid two, so this is invalid, we will have equivalence valid four. There are four valid equivalents in this and three invalid classes on this. So out of this group of invalid classes select any one which is appropriate for that particular system which is in the test or the particular requirement which on the test. Now again we might have one or more covering we can be equivalence class.
It is very important for us to understand that typical requirement behavior also, because we need to see most of the requirements lie with the abound values, right A and B, so surrounding A and surrounding B, how we are going to test it that also we need to consider, which is nothing but the boundary value which is visible to test.
(Refer Slide Time 07:53)



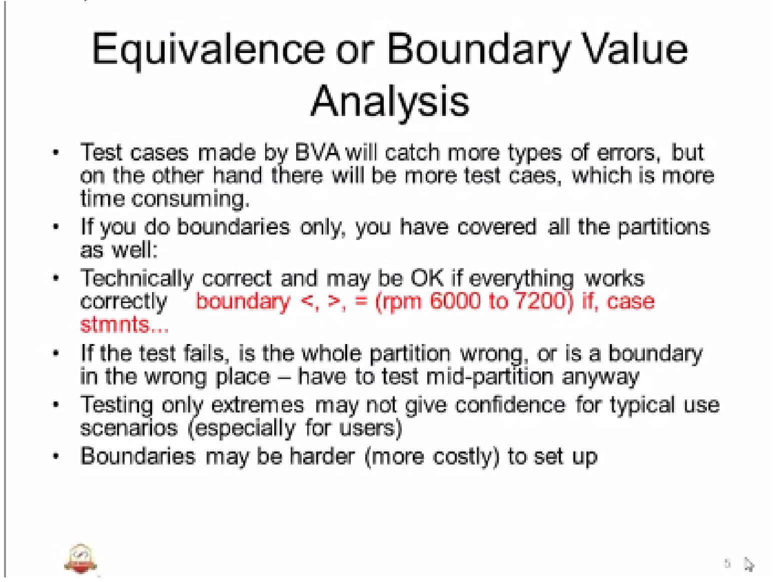
Equivalence or Boundary Value Analysis

- Test cases made by BVA will catch more types of errors, but on the other hand there will be more test caes, which is more time consuming.
- If you do boundaries only, you have covered all the partitions as well:
- Technically correct and may be OK if everything works correctly
- If the test fails, is the whole partition wrong, or is a boundary in the wrong place – have to test mid-partition anyway
- Testing only extremes may not give confidence for typical use scenarios (especially for users)
- Boundaries may be harder (more costly) to set up

5

So continuation of the same equivalence partitioning here we have the boundary values analysis so we choose a selection that is what we have done of each group of equivalence classes' valid or invalid one test case we have selected. We can see that round mark once as the chosen test cases. so if an input condition specifies a range bounded by the, test cases should be designed with value A and B and also just above A, and just below B, like this we are going to have it so the next slide you can see there is a boundary of A just above A and just above B, suppose that is lying on the boundary.
We need to test with the value which is just above A, below A, this is what the first equivalent class, similarly for other equivalence class we need to have it similarly for invalid class also you need to have it in the same way. So which is nothing but the boundary value analysis so the first stub is creating a test cases test boundaries of equivalence classes? For each identified boundary the input and output create two test cases that mean we know that two dots are here for the boundary conditions.
Two test cases definitely are going to be there, one test case on each side of the boundary but both as close as possible to the actual boundary line. That is how the system behave is for the boundary inputs one for the low, one for the high. Or it could be one for the negative the other one could be positive other one for the above and other one for the below, so likewise we are going to have boundary values defined so I will read it again, we have a valid and invalid cases in the partitions and we are going to define them as equivalent cases valid and invalid.
We have that criteria using that each one good selection we are going to have it using the valid and invalid cases, and once we have that we need to add further for the requirement which are lined with the boundary values, because of the boundary inputs and outputs, so you can see for the each of the equitant cases and we have a boundary conditions defined. Okay, the next one is the next cases made by the boundary value analysis will catch more types of error.
You know why, because the system behavior of the boundary could change, suppose to take the variable of 1.0 to 5.0 and we are trying to test 4.9 or 5.1 the system should be held exactly, so we should be predictable with that spellings the chance of catching the errors are issued at the

boundary are more actually, so it can have boundary value analysis, but there is a disadvantage that we will have more aspects on the other hand, there will be more test cases, which is more time consuming, because we need to rectify it, but it is equally important the boundary value analysis test cases, if you do boundaries only, all the partitions which it's definitely going to cover all the partitions but, there is to have a intermediate value for all the partitions but, what is insisted here is you can correct the all partitions of the boundary value analysis is selected, so whether to have 1 to 10 as the requirements for the input, it is good to have a boundary value for 1 and 10 and then intermediate value such as 5 so, how the test cases we will have.
(Refer Slide Time: 13:05)

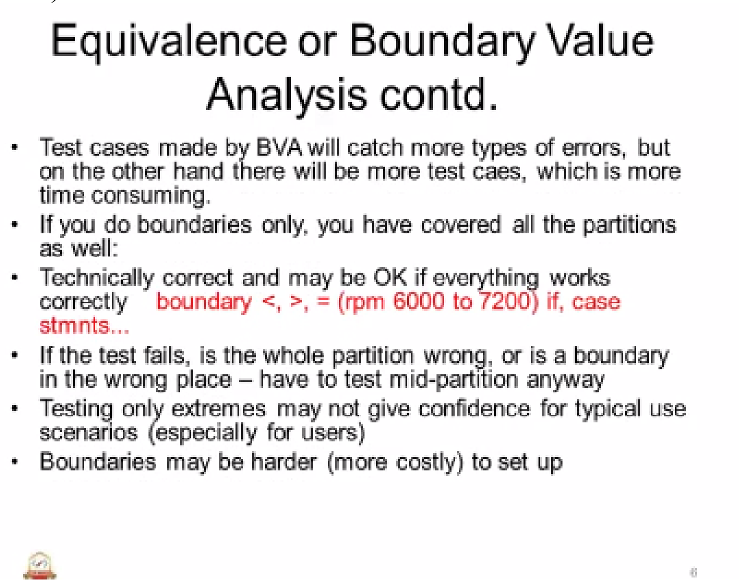


Equivalence or Boundary Value Analysis

- Test cases made by BVA will catch more types of errors, but on the other hand there will be more test caes, which is more time consuming.
- If you do boundaries only, you have covered all the partitions as well:
- Technically correct and may be OK if everything works correctly boundary <, >, = (rpm 6000 to 7200) if, case stmnts...
- If the test fails, is the whole partition wrong, or is a boundary in the wrong place – have to test mid-partition anyway
- Testing only extremes may not give confidence for typical use scenarios (especially for users)
- Boundaries may be harder (more costly) to set up

For 1 we have a upper boundary and lower boundary as 9 and 11 it will become 4 and then the intermediate value that is nothing but the things to move on, totally there are test cases from 1 to 10, if the test fails then the whole partition is wrong, or is a boundary in the wrong place, the test outcome or the test results that, what we are trying to do is to put back the party, that helps to test mid partition anyway, and the testing only extremes may not give confidence for typical use scenarios, which is for the users, ensure that the average intermediate value like 5 when you considered as the range as 1 to 10 usually the system from 1 to 10, it could be 2 or 3 whatever it is so better to identify what is the value.
A good example is the cars sped between 0 to 120, but there is less chances to get, the driver always at the 0 speed and at very few seconds that the driver drives more than 150 or 200 kilometer speed, so trying it or testing it once that the boundary conditions are good, it is very important that the car is stable, and driving in the range around 40 to 60, so that is called as the normal test scenarios, that really gives the input components to the particular distance.
Sometimes it may be difficult to get the stable, because we know that the car speed is 0 to 120 but going by the theory we may not be able to restrict that right, so how will you going to fix it, are you going to take it in reverse, it is definitely not going to be -5 or -10 for sure, but we should have some simulation or some mechanism which is going to be cost deliver.
Then why we need it is low in the terms of cost, theoretically if the speed is less than 0, are likewise the RPM it could be the engine RPM, which goes beyond certain level, and what is the

behavior of that, so likewise we has to have this thing, to test that it may be little challenging in terms of selecting the environment, so we have to plan it accordingly, so basically boundary value analysis is the requirement of equal transaction instead of choosing any resident of the equivalent stuffs, mainly it is focused on the boundaries for each of the clamps.
So, the idea is to select the one test case for each boundary equivalent stuff, the properties of the test case is belongs to the defined equivalent parts, it has the value, which is preferable on one good value we need to select on the clamp, which is close to the boundary of that particular equivalent stuffs.
So the main reason why the boundaries are important is that, they are generally used by program also, and also the execution of the program, so program also implemented the boundary values has a boundary conditions less than greater than equal to so, if that is say the RPM is 6000 to 7200 definitely he has considered the lowest value as well as the highest value, while implementing the code, so that is we need to have the boundary value analysis, so to control the code which are implemented, could have the control of execution is under bandwidth.
So, definitely he will have for implementing this, if the case statement all is implemented I the boundary value analysis, so while doing this, it is bound to happen that he could may be met some mistake while implementing so we will study that what are those.
There are the two sets of the border and while we test the boundary in the both this equivalent facts, so all this aspects is to be considered.
(Refer Slide Time: 19:26)

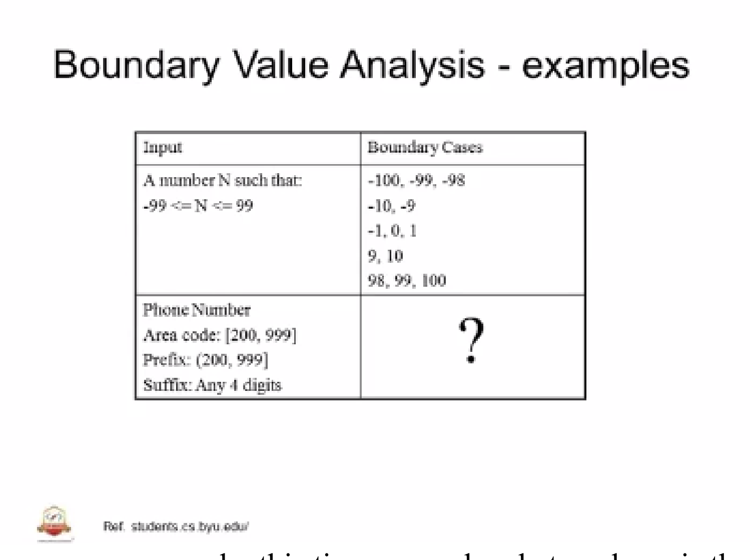


Equivalence or Boundary Value Analysis contd.

- Test cases made by BVA will catch more types of errors, but on the other hand there will be more test caes, which is more time consuming.
- If you do boundaries only, you have covered all the partitions as well:
- Technically correct and may be OK if everything works correctly boundary <, >, = (rpm 6000 to 7200) if, case stmnts...
- If the test fails, is the whole partition wrong, or is a boundary in the wrong place – have to test mid-partition anyway
- Testing only extremes may not give confidence for typical use scenarios (especially for users)
- Boundaries may be harder (more costly) to set up

Okay, so in other terms we will detail out the equivalence or boundary value analysis, the idea behind this principle is, that defects can be caused by the simple program error which is related to erroneous use of boundaries, and so typically the programmer has coded less than or equal is to be coded.
When determining the test cases, values around these boundaries are chosen so that each boundary is tested with a minimum of two test cases, that means the lower on and the upper one, one which the input value is equal to the boundary, and the one that is just beyond it.

So this way we are going to design the test cases under the use of implementation, if a requirement says A < B and in code its implanted as A <= B then the defection possibility is more with BVA than EP, BVA means boundary value analysis, EP is equivalent partition.
So, it will bring out the issue of the boundary values when that implementer has implemented wrong, another example, if A < B is implemented wrongly as A > B then both the EP and BVA could defect the error.
So, this is because the whole purpose of the implementation will collapse here, so definitely in any of the test, definitely it going to be caught, the above one is still we say valid but, equal to condition is considered has the wrong one, we caught in the BVA and EP.
So going by the earlier example what we discussed in few session , where 15 =< temperature =< 40 and assuming a tolerance of 0.1 in the temperature values, the boundary values to be selected are 14.9 (invalid), and 15 which is valid and 40also valid but 40.1 is invalid.
So these are the equivalent values along with the boundary values, so what are those invalid values are noted, 40.1 and 15 is normal, 40 is valid, then we would have the temperature which is equal, so suppose there is no equal to n some 15 so we may have 14 has an valid value.
So there is no equivalent for the boundary value, then 40 would have been an invalid boundary stuffs, and of course we have some 0.1 definitely then we need to have the point only considered the lower side, of applying which, the 25 you can have as well 15.1 here 39.9.
So we saw the valid and invalid values for the boundary value analysis, so these two techniques are very important, then the requirements of the equivalent boundary value we know that, instead of choosing and representing form like the boundary, so that is the main thing in it, okay so it is all about the equivalent and boundary value analysis for the temperature.
(Refer Slide Time: 24:28)



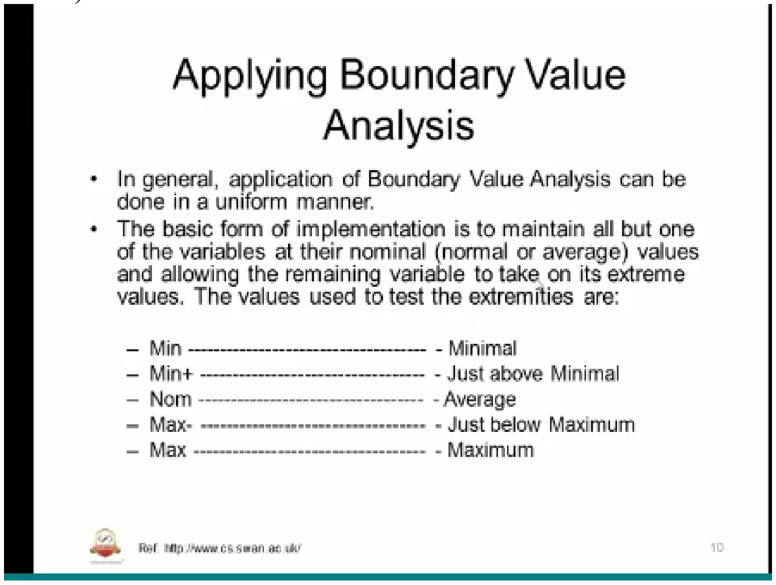
Boundary Value Analysis - examples

Input	Boundary Cases
A number N such that: -99 <= N <= 99	-100, -99, -98 -10, -9 -1, 0, 1 9, 10 98, 99, 100
Phone Number Area code: [200, 999] Prefix: (200, 999] Suffix: Any 4 digits	?

Ref. students.cs.byu.edu/

So let us look into some more example, this time example what we have is the N digit can take it and put from -99 to +99, one of the boundary values, here there is no question of the valid or invalid, so all will have it, but out of which in this cases we have to select it whether it is valid or invalid, so what is the boundary case that is -99 <= N and N if <= +99.

So here are the four cases that we have -100, -99, - 98, of course -99 is the actual boundary case, because it is o the exact edge of the input, similarly we have -10, -9, -1, 0, 1 and 9, 10, 98, 99, 100.
So these intermediate values also considered because these are the valid equivalence of the range, so we need to have this along with the code, and the next example is about phone number. So, we will have boundary cases features area code with 199, 200, 201 at lower boundary and higher boundary 9138, 999 and 1000. The prefix we know that, 200, 199, 198 is the boundary and prefix of higher boundary is with 998, 999, 1000 and we have suffix with we should have 3digits, 5digit test and inputs of suffix.
So, these are all some of the valid equivalent classes with boundary conditions which are allowed to present it. So it is very important aspect of boundary value analysis. So, we have gone through the example,
(Refer Slide Time 27:09)



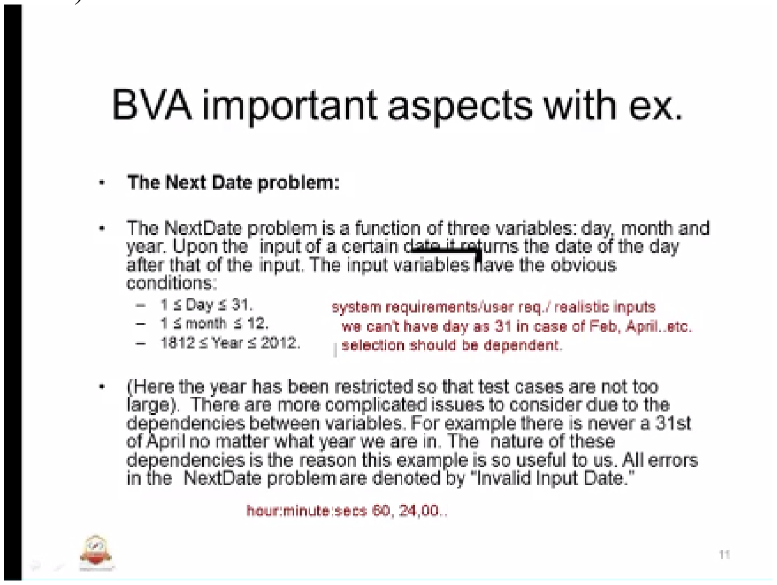
Applying Boundary Value Analysis

- In general, application of Boundary Value Analysis can be done in a uniform manner.
- The basic form of implementation is to maintain all but one of the variables at their nominal (normal or average) values and allowing the remaining variable to take on its extreme values. The values used to test the extremities are:
  - Min ---------------------------------- - Minimal
  - Min+ -------------------------------- - Just above Minimal
  - Nom -------------------------------- - Average
  - Max- -------------------------------- - Just below Maximum
  - Max --------------------------------- - Maximum

Ref. http://www.cs.swan.ac.uk/ 10

In general, application of boundary value analysis can be done in a uniform manner. That means uniformly we can select based on the input, what the requirement is, suppose which requirement is bit complicated and some inputs and outputs then better to have a defined truth table, expected to have a truth table identifying all the combinations first tried to all the combinations then, the first step, the next step is to go for identifying the boundary value analysis along with the equivalent portioning or equivalent portioning it defined or add compliment to the equivalent portioning to be found.
That is what gently behind it. You must give importance for complex requirements. So, that we know that we are going to address the first requirements, each requirements of how it can be treated? The requirement based on requirement to report. We know that how grouping can be done that we have seen in our edit ones. So, basic form of implementation is to maintain all but one of the variables at their normal values and allowing the remaining variable to take on its extreme values. That means normal or average value first we have to see, then we are allowing the remaining variable to take as extreme values the values used to take the values the below,

first is complex requirements which is a type of extreme. So we are going to have a minimal little more than that just above minimum, then nominal that is average or normal.
Then we have a max, maximum value then a maximum value little less than that just below maximum so this is an extremity is limited considered for arriving at forming the boundary value analysis, so this is very important to have it. In one of the terminal you will have this difference from one of the equation in 96.
Boundary value analysis has very important aspects,
(Refer Slide Time 30:20)



BVA important aspects with ex.

- **The Next Date problem:**
- The NextDate problem is a function of three variables: day, month and year. Upon the input of a certain date it returns the date of the day after that of the input. The input variables have the obvious conditions:
  - 1 ≤ Day ≤ 31.
  - 1 ≤ month ≤ 12.
  - 1812 ≤ Year ≤ 2012.

system requirements/user req./ realistic inputs
we can't have day as 31 in case of Feb, April..etc.
selection should be dependent.

- (Here the year has been restricted so that test cases are not too large). There are more complicated issues to consider due to the dependencies between variables. For example there is never a 31st of April no matter what year we are in. The nature of these dependencies is the reason this example is so useful to us. All errors in the NextDate problem are denoted by "Invalid Input Date."

hour:minute:secs 60, 24,00..

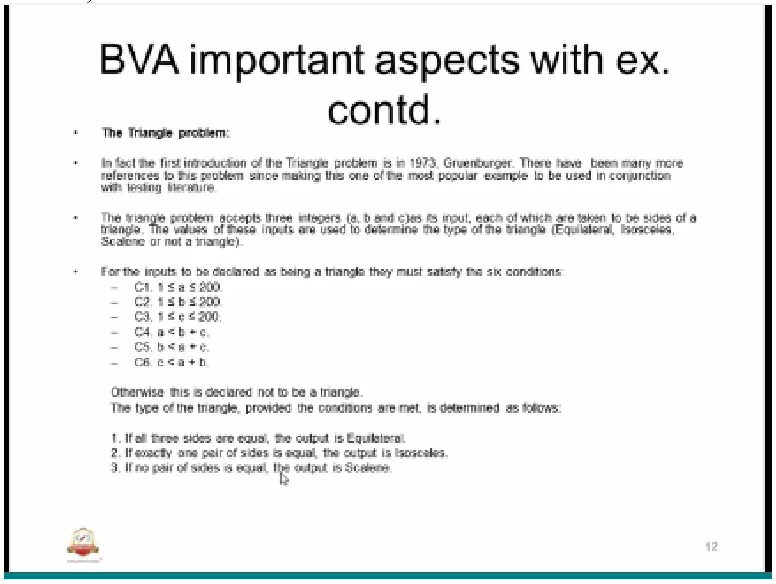
11

The next date problem is a function of three variables: day, month and year upon the input of a certain date of the day after that of the input. The input variables have the obvious conditions like day, it should be 1 to 31, month should be 1 to 12, and year 1812 to 2012. So this is the definition of three variables you know these are some of the realistic. Basically if you have clock also we can derive. So what is the next problem we are going to have? Here the year has been restricted, so that test cases are not too large. Why because? We know that, system can take up to 2012. So, there are more complicated issues due to the dependencies between variables. That means these variables are very important. So that dependencies have to be considered.
For example there is never a 31st of April. That means if we define these dependencies of the date there is only month correct know? If you have a February or if you have a month, so we are not going to have the 31 as an input right? So you cannot have this identifying 31 for the month of April or the month of February or any of the given months. So, no matter what year we are in. The nature of these dependencies is the reason this example is so useful to us. All errors in the next date problem are denoted by "Invalid Input Date" very important thing also we need to understand in that is, we should be aware of the system requirements or user requirements or any what it is called realistic inputs. Why they are important is one value is depending on the dependent on the other one.
Here, we know the month is dependent on the day. So, system which takes the input it is not what you feed, if you feed a month as April, it can have a day of 31. So, we cannot have a day of 31 in case of February or April etc…

So, we need to have a selection in such a way that, selection should be dependent. That means we need to understand what are the circumstances that we came, that we seek obstruct. So that is very important. Not just enough to have those two values just because the pay field takes 31so it also important to understand.

The sub step of dependencies in terms of the variables, similarly let us define a clock so, we have hour, minute and seconds. So this will have dependencies in hour, we cannot go beyond 60, it will go for the next one, similarly we have got 24, that is 24 hours we will have the follower of 01 or 0. Likewise we are going to have the issue of next data column, It is important to have a understanding of the realistic inputs however to feed for the boundary value analysis so you should not get stuck with the column such as next date problem,

(Refers Slide Time 34:58)



BVA important aspects with ex. contd.

- **The Triangle problem:**
- In fact the first introduction of the Triangle problem is in 1973, Gruenburger. There have been many more references to this problem since making this one of the most popular example to be used in conjunction with testing literature.
- The triangle problem accepts three integers (a, b and c)as its input, each of which are taken to be sides of a triangle. The values of these inputs are used to determine the type of the triangle (Equilateral, Isosceles, Scalene or not a triangle).
- For the inputs to be declared as being a triangle they must satisfy the six conditions:
  - C1. $1 \le a \le 200$.
  - C2. $1 \le b \le 200$.
  - C3. $1 \le c \le 200$.
  - C4. $a < b + c$.
  - C5. $b < a + c$.
  - C6. $c < a + b$.

Otherwise this is declared not to be a triangle.
The type of the triangle, provided the conditions are met, is determined as follows:

1. If all three sides are equal, the output is Equilateral.
2. If exactly one pair of sides is equal, the output is Isosceles.
3. If no pair of sides is equal, the output is Scalene.

12

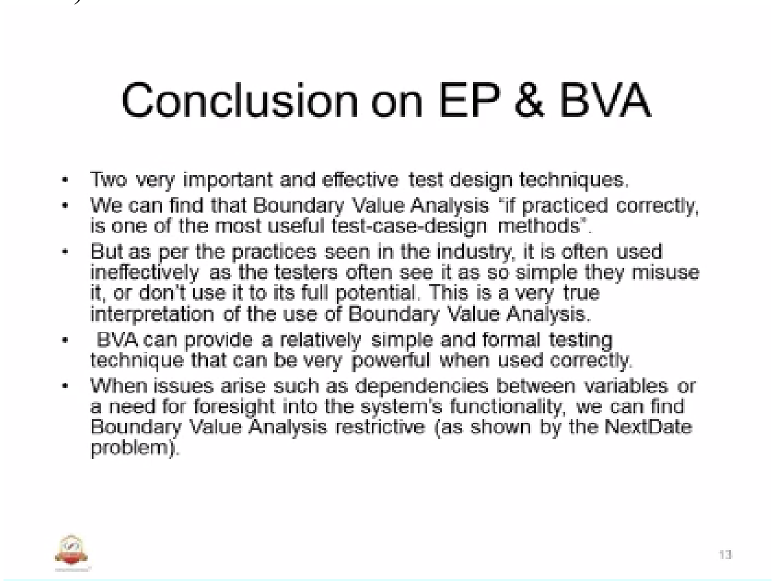
One more is there it is a bit tricky. It is called a triangle problem; I will just go through simplistically, so that it also can be considered so the first introduction of the triangle problem in 73 by Gruenburger. There have been references to this problem since making one of the most popular examples to be used in conjunction with testing literature.

The triangle problem accepts three integers a b c as an input, each of these which are taken to be sides of a triangle that means we have three side's isosolous, equilateral and scalomes whatever you want to call.

So these three inputs are sides of triangle, the values of these inputs are used between the types of the triangle, so what type of a triangle these three inputs are going to decide. So, the triangle could be one of this like equilateral or it could be isolate scalene or it is not at all a triangle so one of this is all factored based on the a, b, c.

So, for the inputs to be declared as being a triangle these must satisfy the various conditions so, c1, c2, c3, c4, c5, c6. So, we have six conditions well before we start the test why because first of all we need to define its being a triangle or not. So, to define that we need to have a triangle defined with condition 1 as a<= 1 >= 200, b<= 1 >= 200, similarly c between 1 and 200.

Then we have other binding conditions (a) should be a< b +c and b< a +c, c< a + b, so these conditions basically should satisfy in order to form the triangle. So otherwise this is never called as a triangle.
So, the type of the triangle provided the conditions or mat is determined as follows:
If all three sides are equal, the output is equilateral.
If exactly one pair of sides is equal the output is isolate.
If no pair of sides is equal, the output is scalent, after we define this triangle we are going to have what type of triangle. So, we need to be very careful in choosing the tests also, we should be realistic in terms of the inputs.
First of all we need to define the criteria how the requirement has been laid out, first you need to understand, and that is what I am always emphasizing that the tester has to have a good knowledge of the system under test. You should not get stuck within the testing and the testing aspects are not good in terms of test selection techniques or test design techniques all that is not good. First having understood the system, we should define testing, the test is when behave in terms of output of the underneath embedded system.
(Refer Slide Time 38:33)

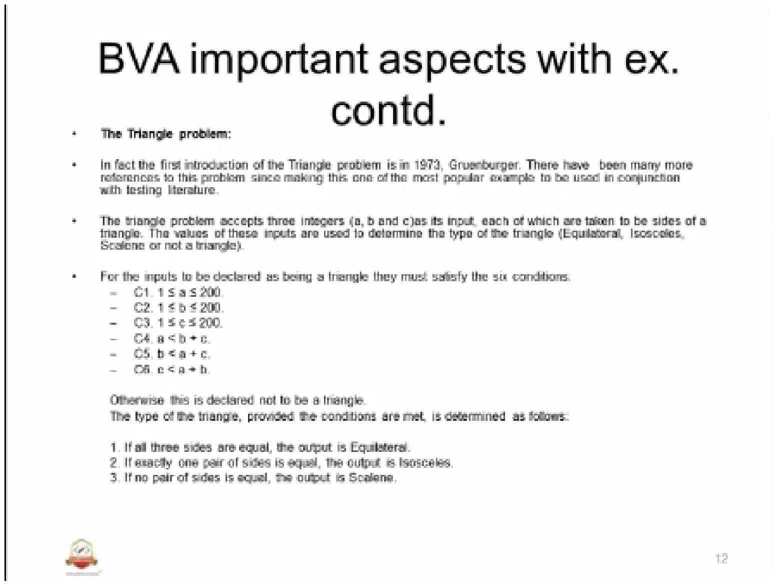


Conclusion on EP & BVA

- Two very important and effective test design techniques.
- We can find that Boundary Value Analysis "if practiced correctly, is one of the most useful test-case-design methods".
- But as per the practices seen in the industry, it is often used ineffectively as the testers often see it as so simple they misuse it, or don't use it to its full potential. This is a very true interpretation of the use of Boundary Value Analysis.
- BVA can provide a relatively simple and formal testing technique that can be very powerful when used correctly.
- When issues arise such as dependencies between variables or a need for foresight into the system's functionality, we can find Boundary Value Analysis restrictive (as shown by the NextDate problem).

13

So, to conclude on the equivalent and boundary value analysis, there are some important points that we need to see is, these two are very effective test design techniques, we have to have a mandatorily includes portioning and boundary values of analysis, we can find that boundary value analysis have to be correctly,
(Refer Slide Time: 39:09)



BVA important aspects with ex. contd.

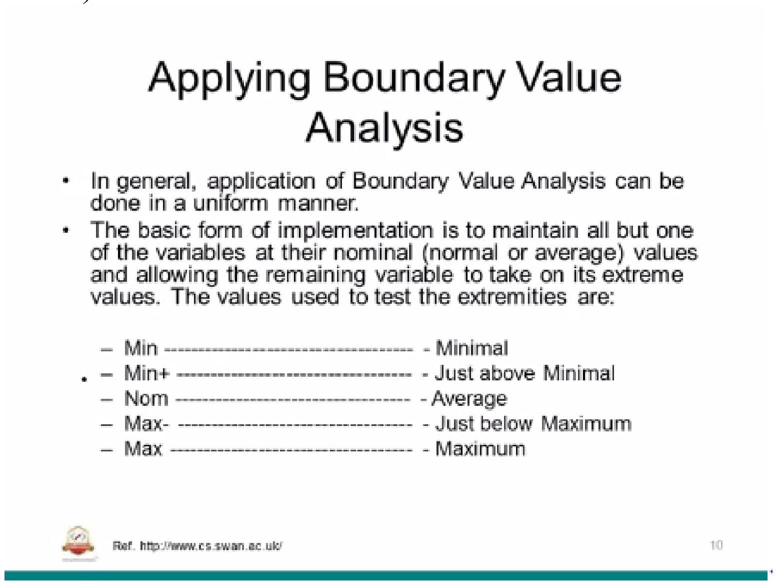
- **The Triangle problem:**
- In fact the first introduction of the Triangle problem is in 1973, Gruenburger. There have been many more references to this problem since making this one of the most popular example to be used in conjunction with testing literature.
- The triangle problem accepts three integers (a, b and c)as its input, each of which are taken to be sides of a triangle. The values of these inputs are used to determine the type of the triangle (Equilateral, Isosceles, Scalene or not a triangle).
- For the inputs to be declared as being a triangle they must satisfy the six conditions:
  - C1. $1 \le a \le 200$.
  - C2. $1 \le b \le 200$.
  - C3. $1 \le c \le 200$.
  - C4. $a < b + c$.
  - C5. $b < a + c$.
  - C6. $c < a + b$.

Otherwise this is declared not to be a triangle.
The type of the triangle, provided the conditions are met, is determined as follows:

1. If all three sides are equal, the output is Equilateral.
2. If exactly one pair of sides is equal, the output is Isosceles.
3. If no pair of sides is equal, the output is Scalene.

12

Practiced or implemented,
(Refer Slide Time: 39:11)

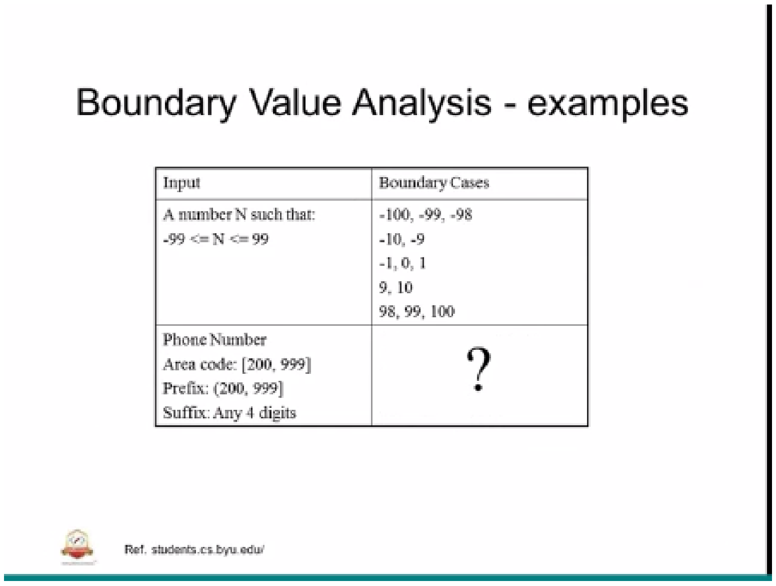


Applying Boundary Value Analysis

- In general, application of Boundary Value Analysis can be done in a uniform manner.
- The basic form of implementation is to maintain all but one of the variables at their nominal (normal or average) values and allowing the remaining variable to take on its extreme values. The values used to test the extremities are:
  - Min ------------------------------------ - Minimal
  - Min+ ---------------------------------- - Just above Minimal
  - Nom ---------------------------------- - Average
  - Max- ---------------------------------- - Just below Maximum
  - Max ---------------------------------- - Maximum

Ref. http://www.cs.swan.ac.uk/ 10

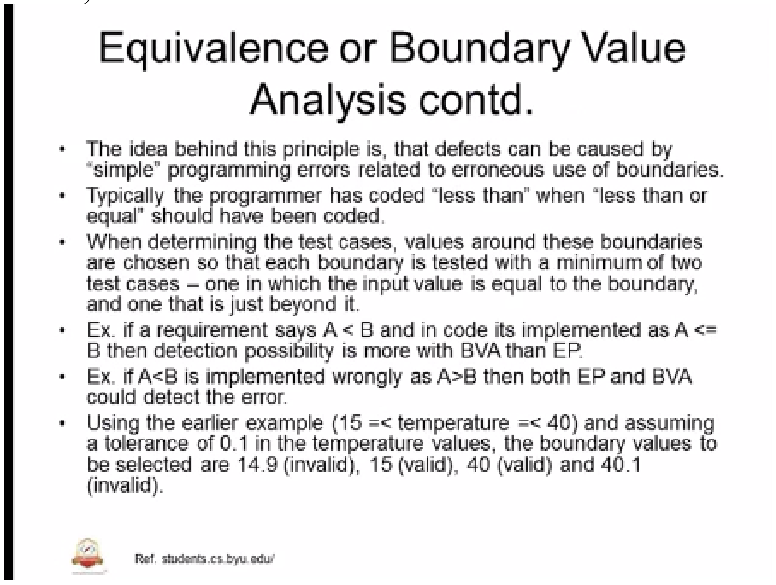
That's why all this issues,
(Refer Slide Time: 39:13)


Boundary Value Analysis - examples

Input	Boundary Cases
A number N such that: -99 <= N <= 99	-100, -99, -98 -10, -9 -1, 0, 1 9, 10 98, 99, 100
Phone Number Area code: [200, 999] Prefix: (200, 999] Suffix: Any 4 digits	?

Ref. students.cs.byu.edu/

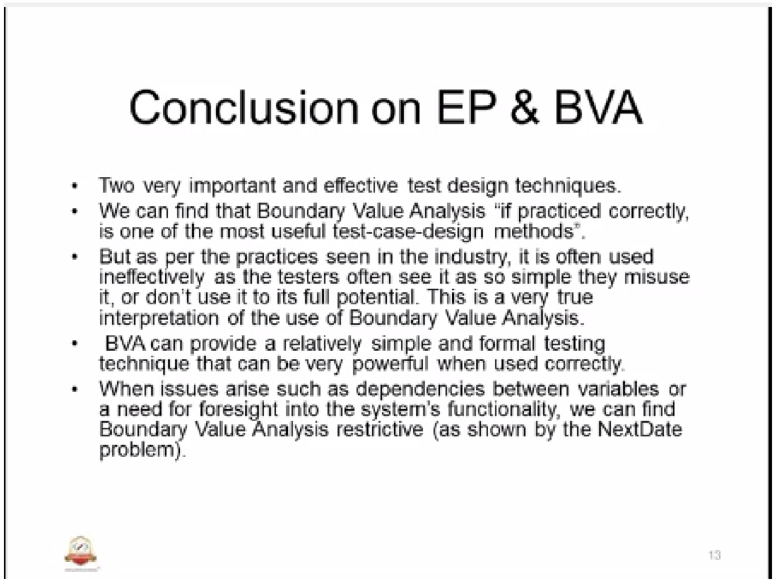
Considering all the ranges considering the different examples,
(Refer Slide Time: 39:15)


Equivalence or Boundary Value Analysis contd.

- The idea behind this principle is, that defects can be caused by "simple" programming errors related to erroneous use of boundaries.
- Typically the programmer has coded "less than" when "less than or equal" should have been coded.
- When determining the test cases, values around these boundaries are chosen so that each boundary is tested with a minimum of two test cases – one in which the input value is equal to the boundary, and one that is just beyond it.
- Ex. if a requirement says A < B and in code its implemented as A <= B then detection possibility is more with BVA than EP.
- Ex. if A<B is implemented wrongly as A>B then both EP and BVA could detect the error.
- Using the earlier example (15 =< temperature =< 40) and assuming a tolerance of 0.1 in the temperature values, the boundary values to be selected are 14.9 (invalid), 15 (valid), 40 (valid) and 40.1 (invalid).

Ref. students.cs.byu.edu/

That you have seen, and kept in is how the boundary conditions it is rare,
(Refer Slide Time: 39:22)



Conclusion on EP & BVA

- Two very important and effective test design techniques.
- We can find that Boundary Value Analysis "if practiced correctly, is one of the most useful test-case-design methods".
- But as per the practices seen in the industry, it is often used ineffectively as the testers often see it as so simple they misuse it, or don't use it to its full potential. This is a very true interpretation of the use of Boundary Value Analysis.
- BVA can provide a relatively simple and formal testing technique that can be very powerful when used correctly.
- When issues arise such as dependencies between variables or a need for foresight into the system's functionality, we can find Boundary Value Analysis restrictive (as shown by the NextDate problem).

13

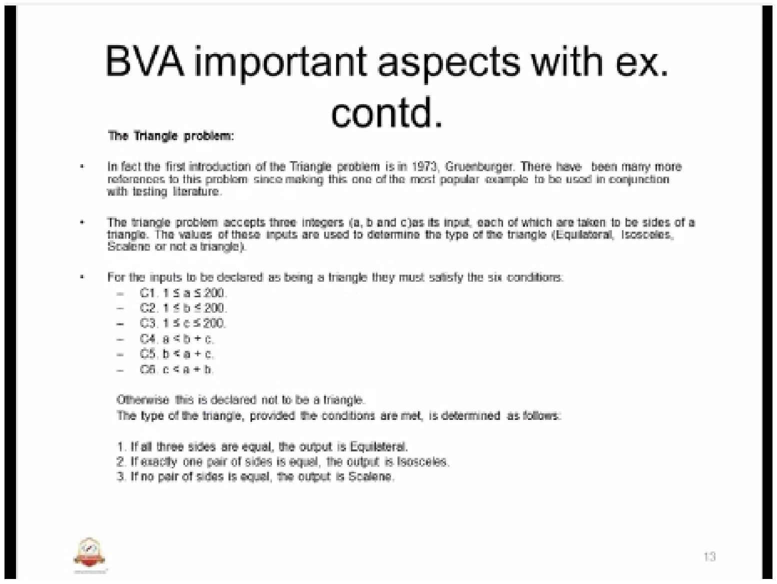
So if it is correctly practiced, it will be very useful in terms of effective, but it will be underneath embedded software, but as per the practice staying in the address field, it is often used the in effectively as the testers often see it as so simple they misuse it, that means I believe that it is going to one, because when I plug it I apply power and it is behaving good and I do the little changes and see if it is good.

As a black box I feel that the system is good and it is contrast, that is not the case, we should not consider data is there primary input. The other we should thing in terms of having a bug and I am going to challenge it in the terms of bringing out all the pieces. Or the test would not have used it full potential, if you get a mobile or handset any telephone instrument, what basically you will do as a user? You will plug it, you will eject on, and you will type to dial, so this is the normal behavior.

So he gets the confident that's fine, but instead of using it at full potential definitely we also think out of the box types of testing it effectively, so for testing it effectively, he needs to understand what it is capable of and whatever the specification, whatever the condition that it can off plug, whatever the technique that he can apply effectively and importantly. So that will bring out all the test defects which are having the issues in the term of implementation, and this more vital are important, why, because the boundary values analysis will bring the effective way of bringing out the bugs.

So BVA can provide relative equivalent formal testing that can be very powerful when used correctly, that means we start formally with the simple technique of the identifying equivalent flashes, and identifying the boundaries of each of the inputs that requirement or the requirements identify, then select the test based on the inputs, select the criteria apply the test and find the errors. When issues are raises such as dependence between variables or a need for foresight in to the systems functionality, we can find boundary value analysis restrictive, that means example take it as next date problem.

(Refer Slide Time: 42:05)



BVA important aspects with ex. contd.

The Triangle problem:

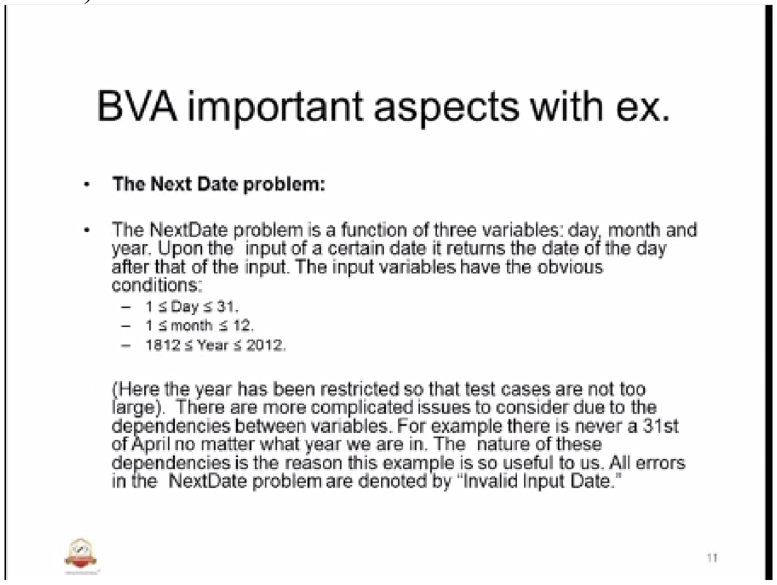
- In fact the first introduction of the Triangle problem is in 1973, Gruenburger. There have been many more references to this problem since making this one of the most popular example to be used in conjunction with testing literature.
- The triangle problem accepts three integers (a, b and c)as its input, each of which are taken to be sides of a triangle. The values of these inputs are used to determine the type of the triangle (Equilateral, Isosceles, Scalene or not a triangle).
- For the inputs to be declared as being a triangle they must satisfy the six conditions:
  - C1. $1 \le a \le 200$.
  - C2. $1 \le b \le 200$.
  - C3. $1 \le c \le 200$.
  - C4. $a < b + c$.
  - C5. $b < a + c$.
  - C6. $c < a + b$.

  Otherwise this is declared not to be a triangle.
  The type of the triangle, provided the conditions are met, is determined as follows:

  1. If all three sides are equal, the output is Equilateral.
  2. If exactly one pair of sides is equal, the output is Isosceles.
  3. If no pair of sides is equal, the output is Scalene.

13

Realistic boundary value,
(Refer Slide Time: 42:06)



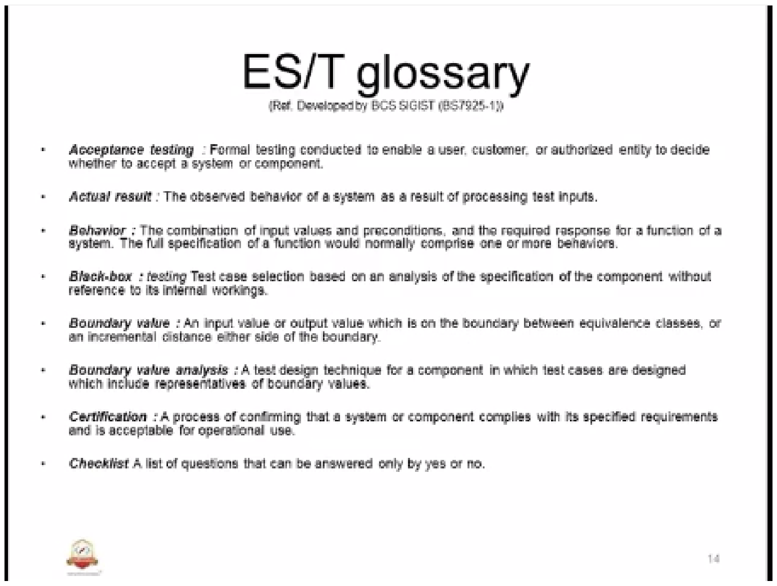
BVA important aspects with ex.

- **The Next Date problem:**
- The NextDate problem is a function of three variables: day, month and year. Upon the input of a certain date it returns the date of the day after that of the input. The input variables have the obvious conditions:
  - $1 \le \text{Day} \le 31$.
  - $1 \le \text{month} \le 12$.
  - $1812 \le \text{Year} \le 2012$.

  (Here the year has been restricted so that test cases are not too large). There are more complicated issues to consider due to the dependencies between variables. For example there is never a 31st of April no matter what year we are in. The nature of these dependencies is the reason this example is so useful to us. All errors in the NextDate problem are denoted by "Invalid Input Date."

11

Should allow for the user to give, once if they free and they help the 31, but this is not a realistic input for an embedded system having a day and the implemented functionality, so we cannot expect that, so there is a restriction for boundary area. We should take a call in the terms of boundary value or taking out such case for such issues, so that is the conclusion of equivalence partitioning the boundary value of his,
(Refer Slide Time: 42:43)



ES/T glossary

(Ref. Developed by BCS SIGIST (BS7925-1))

- ***Acceptance testing*** : Formal testing conducted to enable a user, customer, or authorized entity to decide whether to accept a system or component.
- ***Actual result*** : The observed behavior of a system as a result of processing test inputs.
- ***Behavior*** : The combination of input values and preconditions, and the required response for a function of a system. The full specification of a function would normally comprise one or more behaviors.
- ***Black-box*** : *testing* Test case selection based on an analysis of the specification of the component without reference to its internal workings.
- ***Boundary value*** : An input value or output value which is on the boundary between equivalence classes, or an incremental distance either side of the boundary.
- ***Boundary value analysis*** : A test design technique for a component in which test cases are designed which include representatives of boundary values.
- ***Certification*** : A process of confirming that a system or component complies with its specified requirements and is acceptable for operational use.
- ***Checklist*** A list of questions that can be answered only by yes or no.

14

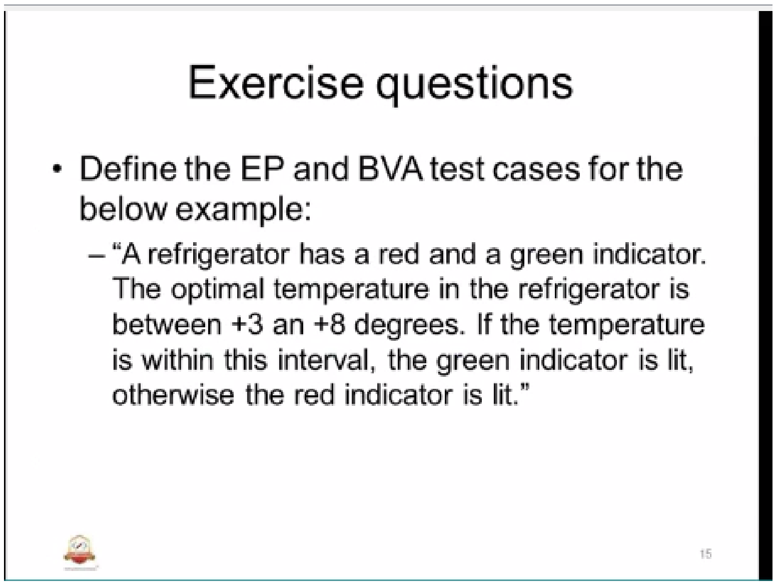
I will have some of the glossary in the embedded software testing add it to each slide in different sessions. So just go through that, acceptance testing. We know that the formal testing from the user perspective actual result, the result behavior of the system as a result of pursuing inputs, you know what is the behavior, the combination of input values are conditions and required response for a function in the system.

So that is what the behavior. The full specification of the function would normally comprise one or more behaviors. You know what the black box is, testing the test case selection based on the analysis of the specification of the components without the reference to its internal knowledge for internal implementation details with all the requirements in the black box. And white box we know about the logic and program work, it is the alphabetical order so that's why it is not interested all that it will use the empty one value futuristic.

So boundary value and input value or output value which is on the boundary between equivalence process or an incremental distance either side of the boundary, it could be upper, below, high, or low etc. boundary value analysis is another testing technique and equivalence class, for a component which the test case are designed includes representatives of boundary values.

Certification, a process of confirming that a system or component compiles with its specific requirements and is acceptable for operational use, so this is the IVM , resources complementary, so we will talk about the certification different process. Checklist, is a list of question is that can be answered only by yes or no. this is basically used by the support of people for the doing the IVNV, it depend validation , verification, mostly this could be done as a tollgate before the product is tested and released, mostly it will be done by the department of test QA.

(Refer Slide Time: 44:59)



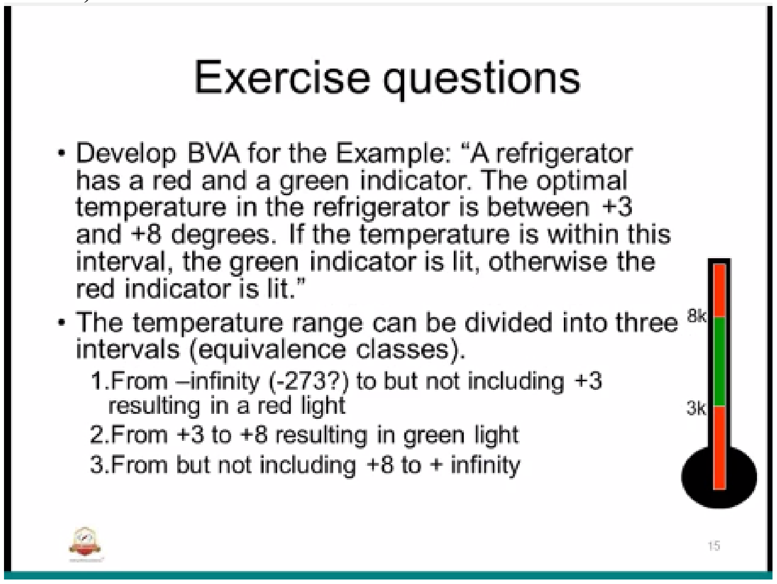
Exercise questions

- Define the EP and BVA test cases for the below example:
  - "A refrigerator has a red and a green indicator. The optimal temperature in the refrigerator is between +3 an +8 degrees. If the temperature is within this interval, the green indicator is lit, otherwise the red indicator is lit."

15

Okay, so we have an exercise question. Define the EP and BVA test cases for the below example: the below examples says, a refrigerator has a red and green indicator. The optimal temperature in the refrigerator is between +3 a +8 degrees. If the temperature is within this interval, the green indicator is lit otherwise the red indicator is lit. that means, the indicator in the refrigerator will indicate a green if the temperature is between 3 and 8, if it is beyond that it will show it as red, so we need to draw a equivalence partition as well as specification of the below.

(Refer Slide Time: 45:54)



Exercise questions

- Develop BVA for the Example: "A refrigerator has a red and a green indicator. The optimal temperature in the refrigerator is between +3 and +8 degrees. If the temperature is within this interval, the green indicator is lit, otherwise the red indicator is lit."
- The temperature range can be divided into three intervals (equivalence classes).
  1. From –infinity (-273?) to but not including +3 resulting in a red light
  2. From +3 to +8 resulting in green light
  3. From but not including +8 to + infinity

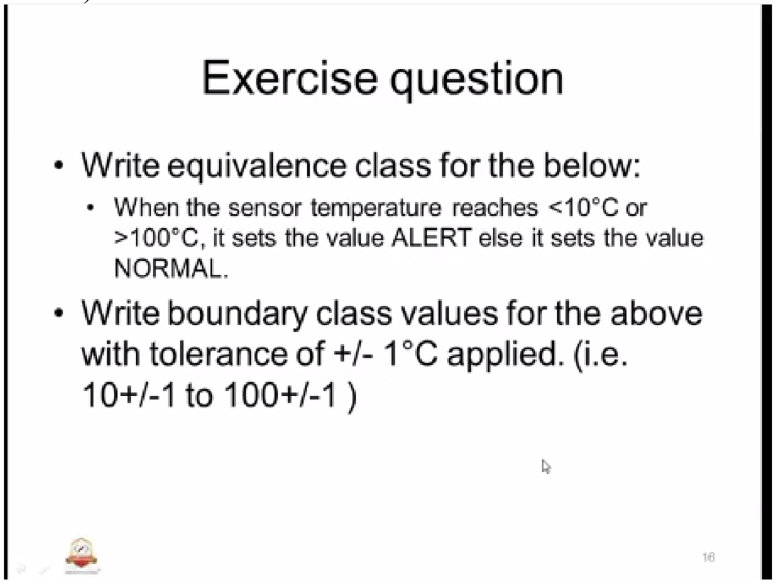
15

This is an exercise, so one more exercise, I think it is the continuation of the previous exercise, you can see the diagram of the temperature indicator. Develop BVA for the example. A refrigerator has a red and a green indicator. The optimal temperature in the refrigerator is between 3 and 8 degrees. If the temperature is within this interval, the green indicator is lit, same thing; the temperature range can be divided into three intervals.

So I just give few inputs accordingly you can define the boundary value analysis and the equivalence partition. So from the infinity it could be minus any value but without including this

3 which will result in a red, here down can see 3 and below are red, and 3 above red, between 3 and 8 it is green. It is the promoter or refrigerator temperature indicator. From 3 to 8 is green, but not including 8 to the higher value is,

(Refer Slide Time: 47:00)



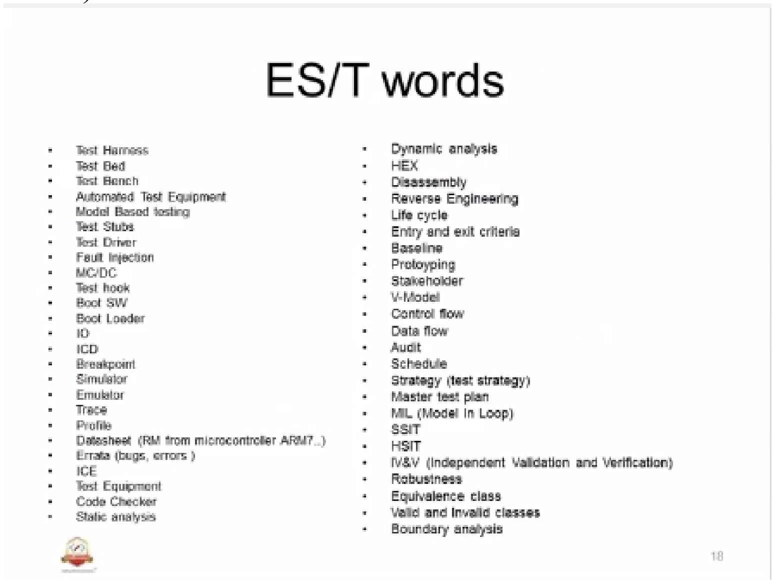
Exercise question

- Write equivalence class for the below:
  - When the sensor temperature reaches <10°C or >100°C, it sets the value ALERT else it sets the value NORMAL.
- Write boundary class values for the above with tolerance of +/- 1°C applied. (i.e. 10+/-1 to 100+/-1)

16

So one more exercise, I think x rays is sets given in the equivalence partition. I will give extension for this, for write equivalence class for the below: the sensor temperature reaches less than 10 degree or greater than 100 degree it sets the value ALERT else it sets the value normal. So write boundary class values for the above with the tolerance, with the tolerance of plus or minus 1 degree, that means, less than 10 degree plus or minus one greater than 100 degree plus or minus 1 it should be applied for doing the boundary value.

(Refer Slide Time: 47:42)



ES/T words

- Test Harness
- Test Bed
- Test Bench
- Automated Test Equipment
- Model Based testing
- Test Stubs
- Test Driver
- Fault Injection
- MC/DC
- Test hook
- Boot SW
- Boot Loader
- IO
- ICD
- Breakpoint
- Simulator
- Emulator
- Trace
- Profile
- Datasheet (RM from microcontroller ARM7..)
- Errata (bugs, errors)
- ICE
- Test Equipment
- Code Checker
- Static analysis
- Dynamic analysis
- HEX
- Disassembly
- Reverse Engineering
- Life cycle
- Entry and exit criteria
- Baseline
- Protoyping
- Stakeholder
- V-Model
- Control flow
- Data flow
- Audit
- Schedule
- Strategy (test strategy)
- Master test plan
- MIL (Model In Loop)
- SSIT
- HSIT
- IV&V (Independent Validation and Verification)
- Robustness
- Equivalence class
- Valid and Invalid classes
- Boundary analysis

18

Okay, some of the embedded system words are, we will go through, we have defined all this in the previous session, we are going to have hamates, equivalence class, word analysis, in your words we have learned today, see in the problem next it tell problem is okay, nominal average

normal will add it. Okay, so we will add nominal, normal average all these meaning are same, it is in the type of input that is there in the system.
So that is about word way analysis, so it is very important to learner, the boundary value analysis and equivalence partition, because entire test is technique, it is founded with the help of these tool techniques, and we could have this for the embedded software testing.