

Hello, hi all welcome you to the next session on the embedded software testing method, unit 2.
(Refer Slide Time: 01:00)

Black Box Testing

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



2

Today we study about equivalence and partitioning, in the last session we had gone through different types of testing just a recap, of whatever methods to have done.
(Refer Slide Time: 01:18)

End of EST Unit1

- Recap of 10 sessions
 - Session 1: Embedded software testing introduction, Embedded system basics, embedded c intro, definitions
 - Session 2: Testing definitions, Role of testing, key processes, test processes, Embedded system test methods and levels, types of testing definitions of Acceptance testing, System testing, Integration testing and Component (unit) testing, what is white-box, black-box testing, regression testing
 - Session 3: Embedded system testing test case design and procedures definition, test standards, test philosophy, what is V&V, debugging, test planning, example test plan



1

Last session we went through any recap, of unit: 1.
(Refer Slide Time: 01:26)

End of EST Unit1 contd.

- Recap..
 - Session 4: Test specification (test cases), example test spec, test procedure example
 - Session 5: Test standards example, example test cases considering a sample set of SRS for EUI (Embedded Unit Instrument) modes of operation, example test case scenario(procedural)
 - Session 6: Levels of testing – unit, integration and system depiction, test harness, EST test setup, host and target based debugging and testing, Simulator, Emulator, target monitor, EST tools set and example list
 - Session 7: T-Emb method definition, Overview, LITO principles, T-Emb lifecycle, techniques, Infrastructure, Organization, commercial tools categorization and a snapshot example



2

All the limitations we have gone through and also.
(Refer Slide Time: 01:29)

Verification) concepts

- | | |
|---|---|
| <ul style="list-style-type: none">• Verification:<ul style="list-style-type: none">– "Are we building the product (software) right?"– The process of demonstrating that the product built is right– It is a process that is used to evaluate whether or not a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase.– Verification can be in development, scale-up, or production which is often an process based.– Examples of Verification is Module Testing, Requirements Based Testing, Integration Testing, etc. | <ul style="list-style-type: none">• Validation:<ul style="list-style-type: none">– "Are we building the right product (software)?"– The process of demonstrating that this is the right product.– It is the process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements.– This often involves acceptance of fitness for purpose with end users and other product based.– Example of validation is the system testing carried out by the end user who is using the product. |
|---|---|

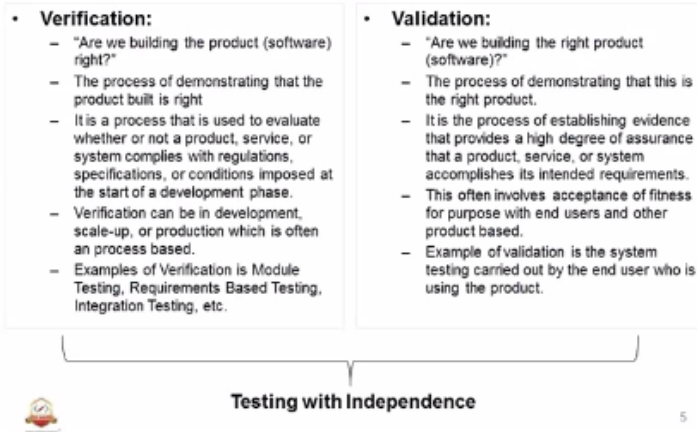
Testing with Independence



5

We defined about.
(Refer Slide Time: 01:30)

IV&V (Independent Validation and Verification) concepts



IV & V validation and verification
(Refer Slide Time: 01:35)

Dynamic Testing

- **Definition 1:** *A process of evaluating a system or component based on its behavior during execution.*
- **Definition 2:** *Testing by executing the program with real inputs.*
- **Other method in context: Static testing** which is *Testing without executing the program. This includes software inspections and some forms of analysis.*

And need to study about or dynamic testing, we have to find subtle definition software. Like in to any report and understand it is function of evaluating system. When system is executing, we do the testing. When we execute the program, another method to compare the dynamic cutting user static testing, this is without the need to execute the embedded software system so okay, we studied about dynamic testing, with them with a couple of definitions also, we know the other method of testing, using the static, which do not need the program to execute. So this could be a software analysis walk through inspections that is all.

(Refer Slide Time: 02:40)

Dynamic testing contd.

- In Explicit dynamic testing, most of the system's functions are tested by means of test cases specifically designed for that purpose.
- In Implicit dynamic testing, during the test process and executions the metrics are analyzed to conclude on the specific testing scenario.



7

Also we have gone through, two types of dynamic testing in a broader level, explain the dynamic testing, and were most of the systems tested. So, specifically for the purpose implement dynamic testing, we do the testing games on, what we have done using existing to the dynamic testing, the process that we have done for dynamic testing will help us in understanding or analyze and support cases all are tested.

(Refer Slide Time: 03:23)

Dynamic testing : Structured basis testing

- Structured approach such as T-Emb method. T-Emb method uses LITO (Lifecycle, Infrastructure, Techniques and Organization) principles.
- Similarly any other testing approaches to align with the test process well defined in the beginning of the EST.



8

And structured basis for dynamic testing, we had already gone through in the uniform that is T-emb methods we can define different aspects. So, similar approaches which we have defined in the beginning of the principles.

(Refer Slide Time: 03:46)

Static vs. Dynamic testing

- Two techniques complement each other
- Static analysis were explicitly used in olden days embedded software testing where tools availability and affordability were on stake.



9

Then, we also studied that, both are required because of dynamic testing alone, they are due to completely test the system, the static means of testing also is important, so both are the technical complement. Static analysis were explicitly used in those days have much tools were expected. So, that is why static analysis more compared to dynamic testing, nowadays dynamic testing in a more used in automated a lot. Wherever it is not possible or way in to complement with the coverage and all that so, there we will do the static testing.

(Refer Slide Time: 04:38)

Dynamic Testing

- Further divided into two basic groups
 - Black and white box testing
 - Black box techniques
 - White box techniques



10

And also we know that, the techniques are always one two basic groups black box testing 1 white box testing and the black box testing or to the techniques white box techniques.

(Refer Slide Time: 04:58)

Dynamic testing: Black box and white box testing techniques

- Strategy
 - The purpose of testing?
 - The goal of testing?
 - How to reach the goal?
- Test Case Selection Methods
 - How to pickup the test cases per requirement/s
 - Which test scenarios are to be grouped?
 - Are they good representatives of all possible test cases?
- Coverage Criteria
 - How much of code (requirements, functionality) is covered?



11

So, we are going through, correctly three part of testing that, we need to understanding different testing so, we detailed about we know that black box testing, (Refer Slide Time: 05:11)

Black box and white box testing

Test Case Selection Methods

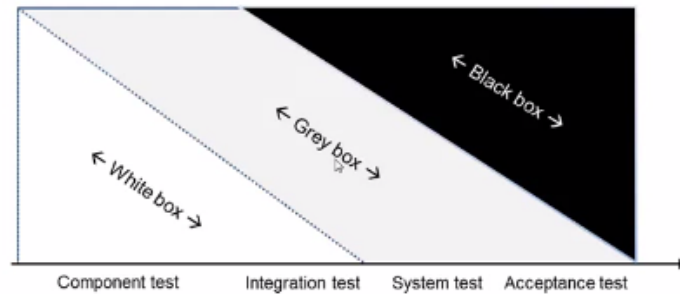
- Black-box / Functional / Data driven
 - Based on the requirements (functional specifications, interfaces, system specification as needed)
 - Black-box test design techniques are based on the functional behavior of systems without having any explicit knowledge of the implementation details. In black-box testing, the component is subjected to input, and the resulting output is analyzed whether it conforms to the expected behavior.
- White-box / Structural / Logic driven
 - Based on the implementation (structure of the code)
 - White-box test design techniques are based on the knowledge of a component's internal structure and uses all the information about how the inside of a unit works, these information might be code and design. White-box tests ensure that each implemented statement is run at least once and are tested against correct behavior.



12

Is basically driven from the data or functionality, the white box, we study detail at the implementation on a logic anything a bit of pair. (Refer Slide Time: 05:26)

Black Box and White Box testing



13

Leave the depiction of a black box and white box you can see the top a black box coming through, functionality white box for a compound testing, see in between certain, embedded systems so, they form a white box with a mix of both white box and black box. (Refer Slide Time: 05:46)

Black box and white box testing: Coverage aspects

- White Box testing techniques measures the code coverage

$$\text{Code coverage} = \frac{\text{Executed Code}}{\text{Total Code}}$$

- Black Box testing techniques measures the features/requirements coverage

$$\text{Requirements coverage} = \frac{\text{Tested requirements}}{\text{Total \# of requirements}}$$



14

And, coming to the coverage we had a CD formula how they do the code coverage, a total code and a executed code both of them are used in coming up how much of the percentage, we are covered so this, is simple a executed code, in terms of percentage is the code coverage, this for the white box of similarly, for black box may be coverage with the requirement, total number of requirements that we have listed, the total number of requirements that will give the coverage. (Refer Slide Time: 06:27)

Black Box and White Box Testing contd.

- **Black Box Testing:**
 - Also called "input/output-driven" testing or "Specification Based Testing"
 - Software is viewed as a black box without bothering about the internal behavior and structure of the program
 - Software is tested against its specifications
 - Test data is derived solely from the specifications
- **White Box Testing:**
 - Also called "logic-driven" testing or "Implementation Based Testing"
 - Permits to analyze internal behavior and structure of the program
 - Software is tested against its low-level specifications or design with knowledge of Code
 - Test data is derived from the logic of the program



15

And also, we had an outlined as a difference between, black box testing and white box testing also.

(Refer Slide Time: 06:35)

Black Box and White Box Testing contd.

- **Black Box Testing Advantages:**
 - The black box tester has no "connections" with the code, and a tester's view is very simple: **code must have bugs**
 - Test cases are designed as soon as the specifications are complete
 - There is need of having detailed functional knowledge of system to the tester and its behavior. Tests will be done from a end user's point of view. Because end user should accept the system. (This is reason, sometimes this testing technique is also called as Acceptance testing)
 - Helps to identify the ambiguity and contradictions in functional specifications.
 - Efficient especially on Larger and complex systems



16

Black box testing advantages

(Refer Slide Time: 06:40)

Black Box and White Box Testing contd.

- **Black Box Testing Disadvantages:**

- Testing with every possible inputs is unrealistic because it would take a inordinate amount of time and tedious.
- Doesn't care about structural and decision coverage, i.e not related to Code
- Exact point in the code, where the software malfunctioning is being done cannot be detected (Code inspection is required to detect the faulty code segment).
- Reasons for intermittent failures of the code cannot be determined (Also called point-to-point Testing, since it just verifies, whether user required functionality is implemented exactly or not).
- Some of the errors may not be able to discover until white box testing is done.
- Heavy dependency on the test environment and stable system.



17

Black box testing disadvantages
(Refer Slide Time: 06:43)

Black Box and White Box Testing contd.

- **White Box Testing Advantages:**

- As the knowledge of internal coding structure is prerequisite, it becomes very easy to find out which type of input/data can help in testing the application effectively.
- Helps in removing the unwanted lines of code, which can bring in hidden defects.
- Helps in determining the faulty code segments.(Ex: use of '=' instead of '==', variable value roll-out problem).
- Discrepancies between the code and the Requirements can be identified, accurately.
- No much System dependency unless there are signals that are derived from the external systems.



18

White box testing advantages
(Refer Slide Time: 06:47)

Black Box and White Box Testing contd.

- **White Box Testing Disadvantages:**
 - As knowledge of code and internal structure is a prerequisite, a skilled tester is needed to carry out this type of testing, which increases the cost in general.
 - It is time consuming as to test all the aspects of the code, covering structure of code and its implementation as per preset standards.
 - At time, there is a need of system knowledge for the internal details of implementation esp. the tolerances, states etc.



19

So

(Refer Slide Time: 06:48)

Black Box Testing

- **Test selection criteria**
 - Equivalence Partitioning
 - Boundary value analysis
 - State or event transition
- Normal & Robustness conditions



20

This are some of the testing aspect now, going to the today session , elaborating, different black box testing methods, so we will detail on this selection the technique called equivalence partitioning what it is called partitioning.

(Refer Slide Time: 07:16)

Black Box Testing

- Test selection criteria(technique)
 - Equivalence Partitioning
 - You have to select a relatively small number of test cases to actually run



2

Okay.

(Refer Slide Time: 07:17)

Equivalence Partitioning

- Background:
 - Typically the universe of all possible test cases is so large that you cannot try them all
 - 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

To understand what is equivalence partitioning, we need to have some background we all know that you do embed systems, certainly with the help of, we have started to study, test planning defining the test cases in given a requirement, or the acceptations, typically the universe of all possible test cases so larger the proposal, we had gone through, example, they want ten so, we know that that system can take from one to ten we know that we can define the inputs, in terms of 1 to 10, but in other systems may have 1 to 10,000 or may be 1000 or whatever so, it bound to happen such that, it may become very larger,

(Refer Slide Time: 08:33)

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



so, it is possible to test all the need so, need to select only relatively small number of test cases, because, we know that it is going to work for one is one two one four one two three ten and have some cases require, so we need to understand what our inputs that basically take, so we need to get them in small number of test cases, in order to be required, so that is what the two-partitioning a number of test cases.

Is whatever defines so it is called impacts, equivalence classes? So, the next comes a question which test cases, we should choose on, which the test case select so for all copies we come across a concept called equivalence partitioning, this will help beginning this test case selection okay, some more details of two test cases are, considered to be equivalent expect the program to process them that means, do you know for a system.

If you feed 1 or 2 or 3, whatever it is, and we get the same result and we do not need to repeat them, we call that as equivalent, similarly we group different equivalence test cases, they all can be grouped how they can be grouped, we will study in the next session.

(Refer Slide Time: 10:21)

Equivalence partitioning

- Most fundamental test case technique
- Identify sets of inputs under the assumption that all values in a set are treated exactly the same by the system under test
- Make one test case for each identified set (equivalence class)
- The partition of the input domain of a program such that a test of a representative value of the class is equivalent to a test of other values of the class. This technique involves designing of test cases for testing classes of errors instead of individual errors.
 - » If one test case in an equivalence class detects an error, all the other test cases in the equivalence class are expected to find the same error.
 - » Conversely, if one test case did not detect an error, all the other test cases are not expected to find any error.
 - » "Equivalence partitioning helps to reduce the number of test cases and ensure that software performs the way it is supposed to do for different kinds of input"
- We analyze the specification and try identify all likely equ. Classes.
- A, b, c..



Identifying the sets of a inputs and under the assumption that, all when losing are treated exactly the same by the system under test, when you set, feed all these values the behavior is same, the output is same, we are the table test with these facts, so equivalence partitioning of, the most fundamental respect of, whatever mandatory testing technique, they adapt okay, so we are hitting the, terms of inputs and result that.

All values or related to the behavior is same, the next one is make 1 test case for each identifiers, that means we are indentify sets of inputs, out of that, you identify one testing, partition of the input domain of a program that, is the test of input, the value of the class equivalent test of other values of the classes it is said in detail, saying that the testing of 1, then, the class equal to the other 1. This technique involves design of test cases.

For the testing classes of errors, in to the levels that means, technique basically identify, the design of most appropriate test case, for a test classes which will build in pages or errors, in a group a bit, system under the test, then of in to the errors, that means, if I feed 1, system is behaving faulty way and they putting 2, 3, 4 different way, we are identify a appropriate failures or errors, between 1 to 10.

Suppose, I know that it is going to say, then the most appropriate 1, so we give detail for them, if 1 test case in equivalence class, all other test case in the equivalence class, with that type of error that it finds. Conversely if one test case did not detect an error, all the other tests case are not expected, that means that group will not finding, other test case also, are not find any other, so other definition that, equivalence partitioning helps to reduce.

The number of test cases, and ensures that software performs the way it is supposed to do for different numbers of input, we can have but it is not realistic to have all the inputs. Especially the larger systems where we found number of inputs, in thousands of values so this equivalence partitioning will help, it is easy so equivalence partitioning is one of the black box testing technique, they identify the idea is the input domain.

Can be divided in two number of equivalence, the characteristics of an equivalence class, that all value belonging to the class or directly same in a program, it assume to select only 1, for each equivalence classes because multiple test cases, of the same equivalence class would repeat right,

so this is initially repeated as well the coverage, measured by dividing the number of inputs, we know how much is left that impossible,

So this basically, to the equivalence classes, the workflow when using equivalence partitioning and, try to identify all likely equivalence classes like we do is we analyze the specification and try to identify all likely equivalent classes, so, when doing this it is important to remember that there may be dependencies between different input value, we feed variables, a, b, c that could be dependencies between B and C, etc, so, very possible input values belongs to that variation is there, t we can divide them in to different class.

So the final next step is to check to choose a good representative out of all these equivalence classes to form this case for that equal so that, is a aim of the equivalence partitioning identification okay, in elaboration of that we probably go through a different example, you may understand better.

(Refer Slide Time: 16:34)

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_E178943D.html

5

Okay, so different views up there, based on the differences that we have and put it in the bottom also to have a clear understanding equivalence partition theory and proposed by older is attached to reduce the total number of this test cases density by partitioning being independent entity finite number of equivalence class which we know to reduce the combination.

Or combinatorial explosion of potentially respirators, that means as part of the strategy the next thing is we do a sampling that means, there are 100 possible way to point a book test case or finite so, what we do is? We may sampling like 10 percent 20 percent of that, which is nothing might be atoms or the fables of the particular system, so that we will be done with the help of equivalence partitioning, that is what it means okay.

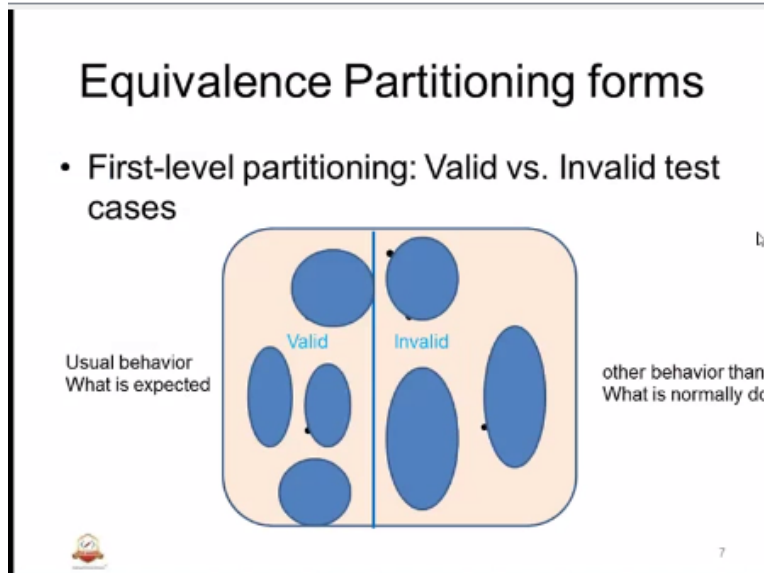
(Refer Slide Time: 18:02)

Equivalence partitioning contd.

- It's a principle of Deriving the test cases, the input domain (all possible input values) is partitioned into "equivalence classes."
- For all input values in a particular equivalence class, the system shows the same kind of behavior (performs the same processing).
- The idea behind this principle is that all inputs from the same equivalence class have an equal chance of finding a defect, and that testing with more inputs from the same class hardly increases the chance of finding defects.
- Instead of testing every possible input value, it is sufficient to choose one input from each equivalence class. This greatly reduces the number of test cases, while still achieving a good coverage.
- Contd..



Continuation of the equivalence partitioning, it is a principle of deriving the test cases, the input domain all possible input values is partially equivalence class, for all input values in a particularly that, shows the process is the same, output that expected of that, or the functionality that execute that most set of equivalence classic input, the idea behind this principle is that all inputs from same equivalent classes, having an equal chance of finding a detector and that testing with more inputs not the same classes increase the chance of finding defects. It would not defect any more design, finding a different effect, instead of testing every possible input value, it is sufficient to choose one input from each equivalence class, this greatly reduces the number of test cases, while still achieving a good coverage , so, what we understand from the equivalence partitioning we divide the entire bunch of all test cases, based on the behaviors, that inputs could provide will result and output will failed in out passes so we divide all of them in equivalent classes.
(Refer Slide Time: 19:39)



So, the equivalence partitioning feels basically on, it is called a two-level of, two partitioning one is called valid other is invalid that means the one that are going to be tested with the help of the team you can say normally or respectively, you will behavioral that is called valid test cases, or very different class usually behavior or what is expected from the customer under the test, all this comes under valid class.

The one that is not so usual or what is not expected, when the system is running normally it should have each will be very ready other behavior, landmark what is normally done. So, it can be classified as, so two groups of relevant, these are all coming under an equivalence partitioning and the equivalence partitioning, we are going to have two types of partitioning valid and invalid probably okay.

so let us see the equivalence partitioning or, what are the types of that we have equivalence partitioning, so there are two basic methods top two basic curve partitioning that we have for equal partitioning, that is why is called as partitioning one is valid partitioning and another is multi-partitioning, so, what is valid partitioning when we have all the system learning normally. That means executing normally here occurred in the field working as expected the test. That will resulting normal behavior or usual those are all coming under the test cases, the inputs that with the help of test cases will result as usual behavior, or the normal, suppose 1 to 10 is what is system accepted in the, or 1 to 10, is what type? the system is expected to taking both and give us a no concern will come at all valid term equivalence, so the other way of testing with the help of negative values or outside the boundary.

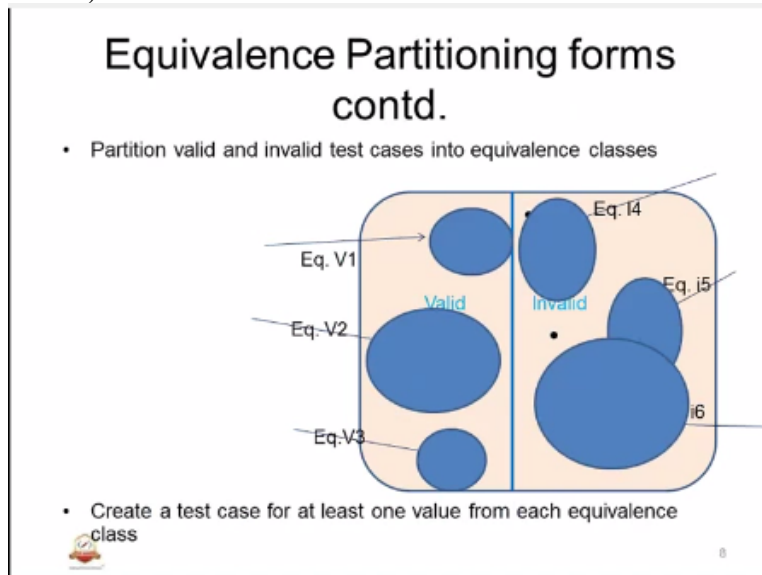
We choose result in some sort of an, what is called? other behavior of the normal system, so those all coming under invalid test case, so basically what we have trying to say, we may have, see left side you can keep number of test cases, dots land dots are all in this test cases we can group them something like one group is parties, other group could be for this, and one more group can comprising, more 3, I will repeat.

So we have this rectangle box identified, the entire group that is what we do? In that we are going to have a two-partition valid, valid, what we do all the valid inputs which you behaving the same output, we are going to define 1 equivalence class, this 1 equivalence class, the first one this

could be, this one the next could be this one, 3 test case so we have four equivalent valid equivalent class, similarly for invalid also.

We are going to have something similar this, something to this, we are having 4 invalid test cases, we are having more actually, but what we do? It would be the behavior is that is why? The equivalent comes under the picture. Equivalent have two things one is valid another one is invalid so, these are the primary first level of partitioning.

(Refer Slide Time: 25:03)



The same diagram, that we have seen in the previous slide here, what we have is we define or we name each of them with a number equivalent 1, equivalent 2, equivalent 3, etc. so further maybe you can change it as something like, equivalence valid 1, equivalence valid 2, equivalent valid 3, similarly equivalent invalid 5, equivalent invalid 6, so we have three valid classes and three invalid classes and we saw in the previous slide

we again we get the group, each of this we do not have it, which are resulting in same behavior, so we have, we are to cover this, the program could be so we have valid 3 test cases, similarly we have invalid test cases, all this test case can be valid or primary valid so, identify all the test cases, and divide the test case in to two, lower level partitioning, what is valid and invalid? So, then we group valid such way that.

The behavior is usual and similar bas going to come all, of these are grouped so there are four here, the test case is 3, so we made it as equivalence valid to work, equivalent 2, equivalent 3, similarly we can name it invalid it also with any kind to different so that is what high light here, partitioning valid people created a test case, for at least 1 valid for each equivalence class, we have to be 3, probably select an appropriate one.

Which is good enough to have the absolutely form, that particulars valid or in valid class. So, valid and in valid test cases, equivalence class first and then create the test case but at least one value of each equivalence class, check what is it?

(Refer Slide Time: 28:15)

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 <i>Similar for Prefix and Suffix</i>



Ref. students.cs.byu.edu/

Let us take an example with the help of or the help of that, probably we are here okay, this is a rectangular box identifies three columns.
(Refer Slide Time: 28:37)

Equivalence Partitioning - examples

Input	Valid Equivalence Classes	Invalid Equivalence Classes
A integer N such that: -99 <= N <= 99	?	?
Phone Number Area code: [200, 999] Prefix: (200, 999] Suffix: Any 4 digits	?	?



Ref. students.cs.byu.edu/

The first one is being an input, suppose you take this an input that means, this an requirement 2 or there, EP here N, that N takes from - 99, so that is what the input it goes for a requirement , so what are the valid equivalence classes, that we can arrive it, like what are the invalid equivalence classes that.

We can arrive it for this particular input sector typically one more, will defer a phone number, phone number will have area code 200 and the prefix is 200, 999, 1846 2014 can be there, test case for the input and what are the valid equivalence classes, and what are the in valid implements classes for this okay, first one we will take, we know that the N is not –the system that respected to take the input – of 99 to +99).

So we can have number of test cases, -99, -98, -97 etc., -99,-98,-97, but all are within the range , then we can have 0, 1,2, 3 10 20 30 50 60 90,99, right so these are all it can take this, what we have but do we need to have all this we know that this are all valid because the first thing is, these are all valid, valid equivalence class, still is not equivalence class, it just a valid, that they can take, -99 take -98 can take etc., up to 99.

So these are all test cases but we do not get to have, probably he can group, this again based on the system what is going to be tested, it could be a unlock or system having a sensor or input, taking different values, we need to understand the system and probably system perspective probably to use that, in general we can choose a group of test cases whether it makes emphasis, here we have 1,2,3,4,5, so all that need to have to be tested.

So this will comes under the equivalence class, this will also became a valid equivalence class, so here about 5 are there -99 -10, so two negative set we want to have with the two digit number, at some more we want to be feed because, the implementation could have the issues that difficult to feed it, so that way we can divide it, second equivalent class -9,-1, where two digit negative number, then that is a value is 0 so here also we should accept. Then similarly we have 1 to 9, 1 and 9 on the single digit set, and have to be 10 and 99, so this is the good example of equivalent class.

Okay, the next one in valid equivalent class, so what could be the in valid equivalent class, we know that valid equivalence class, we have define from test case 99, it is invalid for N, but invalid test cases, but invalid test case could be anything, it can be -99, -98, it could be -100-101,102 etc..It can grow, but better to identify few of them. Similarly, that side we have 99, it can take 100, 101,200, 210, randomly we have to wait.

Those are all will be coming principle equivalence class test cases, but under the category of invalid, why? Because as a normal case, this is not expect to take that, so this is will be coming under invalid test cases, means, it falls on that side, not in usual manner that is why is called as invalid test case, so what are the few example in invalid equivalence classes, -99, > 99, we can test the values -100, < 99, then numbers are all equivalent.

Let us the system will accept the negative, at the usual value, instead of -12, similarly we are giving the input as 12-2, so this required as a value, instead of numbers, so the system which accepts the input, that we should provide this, 12-, similarly 1-2-3, likewise we can add, in terms of numbers are normally but, similarly numeric sign that means instead of number, we are trying to give a characters.

Like, which it should not except, alpha numeric is here, many characters so, maybe you would have see in this, software all need a password right, so password it suppose to expect define input on the, cannot change, so that is the few example of providing input, the input act as same manner, what is specified requirement? It is the lengthy value it is a blank, so the blank has a specific problem, so these are the two valid and invalid equivalent classes.

Similarly a phone number valid equivalent class, we know that area code 200,299, so we can have (555), (555), then 55, 5555, with 5555, we have 55555, 55555, this is a another value phone number, then we have 200, = area code, these are =96, between the value and identify 200 to 99

this is the last one suffixes, identify as 200 to 99, so this is similar to days for the end. Here we have the phone number, which is specified?

So what are the invalid equivalent classes, invalid format is 555, 5555, together, so similarly the next one two prefix all there, suppose if we want one prefix, area code >200, so the 100- value, similarly it could be 999, then other invalid classes is, one itself invalid classes, so the entire group has one invalid class, so we have four invalid classes. So area code with normal characters, such as landline, all this can be invalid equivalent classes, so this is the example of equivalent classes, so we will study more examples in a next class okay.

Now, having this is an example definitions we will again study the equivalence classes, two types of equivalent classes, valid equivalent classes, such as valid inputs with the program, all other inputs, which are not include that, valid class are called invalid equivalent class okay, a few guide lines for identifying the equivalent classes, first if where input condition specifies the values, 1 to 100.

The equivalent classes are, one valid equivalence class is count from 1 to 100, two invalid equivalent class count <1 and count > 100, okay, so the next guide lines, it explains the different sort of techniques that is what is guidelines talks about if an inputs specifics set of values which are handled differently like type of color like red, blue, green it is an input a set of input then the equivalence class what are they valid equality class for each values. That means red and then we have blue then we have green this is on valid equivalence.

(Refer Slide Time: 38:59)

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

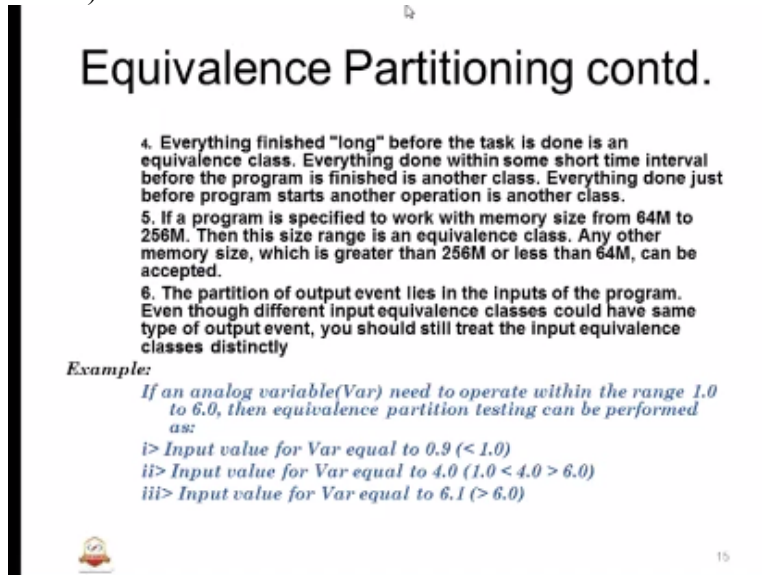
This is one valid equivalence class one more valid equivalence of this aspects may be it could be white, or yellow it could be, if an input conditions specifies a must be value the character must be a letter suppose the equivalence class are here basically depends on the type of implanting requirements, the requirements talks about the count.

The count well have terms of number here is a character so we need to add with that invalid character, invalid group of characters so that blue, green is a valid group other one is a invalid equivalence class similarly we have must be a value in short of that in conduction in available then what are the invalid equivalence classes talks about is any letter should concenter the ere

letter it has is nothing but a valid equivalence class other than ere litters or it is called a non-litter it could be number that is nothing but invalid equivalence class.

So all this are charted here it very simple you can see where we have taken only the numbers as an input and what are the non-numbers all this are part of the non-valid equivalence class and valid numbers all is in the range are called valid equivalence class the fourth guideline.


(Refer Slide Time: 42:12)



Equivalence Partitioning contd.

4. Everything finished "long" before the task is done is an equivalence class. Everything done within some short time interval before the program is finished is another class. Everything done just before program starts another operation is another class.
5. If a program is specified to work with memory size from 64M to 256M. Then this size range is an equivalence class. Any other memory size, which is greater than 256M or less than 64M, can be accepted.
6. The partition of output event lies in the inputs of the program. Even though different input equivalence classes could have same type of output event, you should still treat the input equivalence classes distinctly

Example:
If an analog variable(Var) need to operate within the range 1.0 to 6.0, then equivalence partition testing can be performed as:
i> Input value for Var equal to 0.9 (< 1.0)
ii> Input value for Var equal to 4.0 (1.0 < 4.0 > 6.0)
iii> Input value for Var equal to 6.1 (> 6.0)

 15

Everything finished the long before the task is done an equivalence class. Everything done within a short time interval before the program is finished is another class everything done just before the program starts another operation is another class that means we have a time based requirements suppose in those cases how can define the equivalence class what is talks about so everything finished long before task is done is a valid equivalence class.

Everything is done in some short time interval before the program is finished is anther valid class we can say everything is done just before the program starts another way operation is another class this three equivalence class has ante defined for the time that is one guideline that is the first guideline fifth one it talks about the memory if a program is specified to work with memory size suppose.

The memory can take from 64mp to 265mp so what is going take for equivalence then the size range is an equivalence class because we know the size what is low size and what is the higher size any other memory size is greater than or any other size is greater than 64m it can be accepted those are called invalid equivalence class like ways we can define the equivalence partitioning for this short of a requirements.

The last short of a guideline is the partition of output event lies in the inputs of the program. This is very important so it is not just the input based equivalence partitioning there are output based partitioning also that is possible due the sudden state of inputs the outputs can be derived so based on the outputs also we need to defined that is what it is means even though different equivalence input class could be could have same type of output event,

We should still treat the input equivalence class distinctly what it means is we have two equivalence class but it still it is result in the same output but it is based on the type of

equivalences class we need to categorized that. For example it may show an error for all this as a one error so that means we have one output but still we want to test it because we know what input it can take so that what it means?

So output event is based on that so one example we will take here if an un-log variable were we to operate between the range 1.02 or 6.0 so is an un-log input which can take or it will operate in the range of 1.02,6.0 so what are the equivalence class that can be performed first type of equivalence class input value was variable =.9 that is 1.0 has an input value as a variable = 4.0 this is normal range this 4.0 is greater than 1 and less than 6.

Wrongly put here it is this way so 4.0 is grater then 1.0 and less than 6.0. So that way we can have the second set of equivalence the third one will be input value for the variable = 6.1 which is just outside the range this is also an invalid class so in the help of this guidelines we are defining the invalid and valid class partitioning class okay one more example.

(Refer Slide Time: 46:37)

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

System behavior is subjected to the following condition regarding the input temperature close temperate lays within 15 to 40 here is very important is = in here it is including 15 and including the 40 out of mothering so sometimes we have to be careful in doing it for test case selection because the phase less than those are all prevail that it is not a good practice to have a recruitment then it is less than so what will happen with equal.

So we just do not get confused so it is better to emanation is less or = similarly as a outer boundary as greater then = or lesser then = whatever the way the = do have the greater then less than operates are used the number of possible values for the temperature is huge this we can have 15.01.16.02, 15.1, 15.215.3 like this tell 39.999subject to the expect blitz of the embedded system so it is in fainéant the value could be different.

We know that we are going to have equivalence protestation shake of producing all this so that is what is that it means here three equivalence class can be defined for this temp0ructure is lower the 15 lower then or =40 temperature has a value sorry temperature is lower than 15 because = it is also valid class temperature is only the range of 15 to 40 temperature is higher than 40 so three equivalence class can be defined.

So three test cases can be sufficient to cover the equivalence classes we do not have divisions because it is enough to test it is conduction so invalid two are there 10 and 50 valid we have one it could be anything it could 15 to 40 that is 35 the processor of the particular value depending on the volt exactly we are trying to testing if it temperature of the requirement it is 35 if you think that the system could fail 35 point of a 39.5.

You can have study it. We will study about that go through value and other value in next slide because that will come. Compliment further equal portioning that is also the important technique. We will study about what value on this will be okay, equivalence partitioning continued one more good example I will take it here let us considered the level for which I set an indicator as per the below conditions.

So fuel level definition is the level of the fuel, fuel of the level it will indicate that is the requirements the three requirements it the level goes below 10liters it will set the indicator to yellow. If the level goes above 100 liters or below 1 liters it shall set indicator to red otherwise it shall set the indicator to green so what it means from 10 to 100 liters we have the indicator set S and the above the 100 or below 1 liters it shall set as 1 below 10.

And above red it will set as yellow so it is have the understanding of different values and this should be see the requirement of the as follow itself. The one requirement for the which talks about 10 to 100 liters the 1 talks about that which is less than 1 liters or above 100 liters the indicator will ensure it the 1 that is showing will indicate an yellow so what are valued and what are divided with the help of this below one.

We can identify invalid 0 and 1 the value which is in between or invalid who is indicate the value between 9 and 10 are valid yellow indicator because that requirement talks about yellow. So it is valid the next one the value between 100 or less than that is green because requirement tells that between 10 and 100 should be green and 1 and 10 should be yellow and before 1 it is invalid similarly the last one is select the indicator as the red defining the invalid input is there invalid input could be greater than 100 it is 101 to 10 etc., so that is what it interrupt with the help of this examples.

(Refer slide Time: 52:23)

Equivalence Partitioning example contd.

Levels / Indicator	A) < 1	B) <10	C) >10	D) >100
Red	✓	X	X	✓
Yellow	X	✓	X	X
Green	X	X	✓	X

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

So this is a table this is a good sort of input table which we called the truth table why because it in defines all the combinations of input and which is good enough to identify and differentiate between invalid and valid class so what we have see here truth table identifying on the left hand side you see as I said yellow, green and indicator and it dependences is the different sort of class so what are those class we have, we have class A, class B, class C class D.

Identifying for different indicators less than we have the indicator which is said to be truth table as ticked similarly we have yellow indicated less than 10 may be we can add 1 also it is up to you how you want to design the truth table similarly we have the end indicator coming in green similarly we have greater 100 coming under red we can also add less than 100 greater than 10 some more we can take it as a 10.

So out of all this what are the valid equivalence what are the invalid differences so we have four about groups here invalid 0 and -1 because the requirement does not often 0 and 1-1 embedded design it is a invalid. Similarly other extreme of the requirement is 101 to that terms are invalid equivalence partition, so we have to write one volume 2 equivalence partitioning 1234 up to 9 10 11 12 13 up to 100. So in this valid equivalence we can choose something like 158 and 9 or it could be 1 is not similarly for the valid group of equivalence class you can have a 20, 100 likewise.

(Refer Slide Time: 55:39)

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.


So in this output equivalence also can be done this equivalence partition with done with the help of that means out of this is at partitioning where outputs can be ready for different inputs and signal inputs also then group testing which will come for equivalence partitioning test cases are then derived that cover all equivalent output classes that means it derives all which will take care of all the output classes also.

Similarly we will define like the two tables we have for input for the output also, so with that we come to that the equivalence partitioning so we have gone through some of the type of examples the defined valid and invalid process. And guidelines so that count can us thus could be value and the size time because majority of all the requirements that are come in to this guide lines. Also we went through the input examples.

Then we had defined two partitions valid and invalid they are going to identify and in this state call this as test case transition partitioning and we identified couple of inputs for equivalence partitioning we also know that output into partitioning.
(Refer Slide Time: 57:25)

ES/T words

- Test Harness
- Test Bed
- Test Bench
- Automated Test Equipment
- Model Based testing
- Test Suite
- Test Driver
- Fault Injection
- MCD/C
- 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
- Prototyping
- Stakeholder
- V-Model
- Control flow
- Data flow
- Audit
- Schedule
- Strategy (test strategy)
- Master test plan
- MIL (Model In Loop)
- SSIT
- HSIT
- I&V (Independent Validation and Verification)
- Robustness
- Equivalence class
- Valid and invalid classes
- Boundary analysis




20

In the end of the class we have added more words deriving as the embedded software testing warrants, I think we have everything so in that step what we had added is IV/V, robustness, equivalence classes, we have today valid and invalid classes of course boundary analysis we do in the next class.

(Refer Slide Time: 58:16)

ES/T words

- **What are the main test selection criteria for black box testing?**
- **Write equivalence class for the below:**
 - When the sensor temperature reaches $<10^{\circ}\text{C}$ or $>100^{\circ}\text{C}$, it sets the value ALERT else it sets the value NORMAL.
- **Write boundary class values for the above with tolerance of $\pm 1^{\circ}\text{C}$ applied. (i.e. 10 ± 1 to 100 ± 1)**



20

And we have couple of exercise questions what are the main test selection criteria for black box testing? So we have this, write equivalence class for this requirement? When the sensor temperature reaches $<10^{\circ}\text{C}$ or $>100^{\circ}\text{C}$, it sets the value alert else it sets the value normal, so we have a temperature sensor and alert which goes below 10° or goes above 100° or else it sets the value which says the indicator is normal. Write boundary class values for the above with

tolerance of + or $- 1^{\circ}\text{C}$ applied that is about the class in next session. So that is for end of the session.