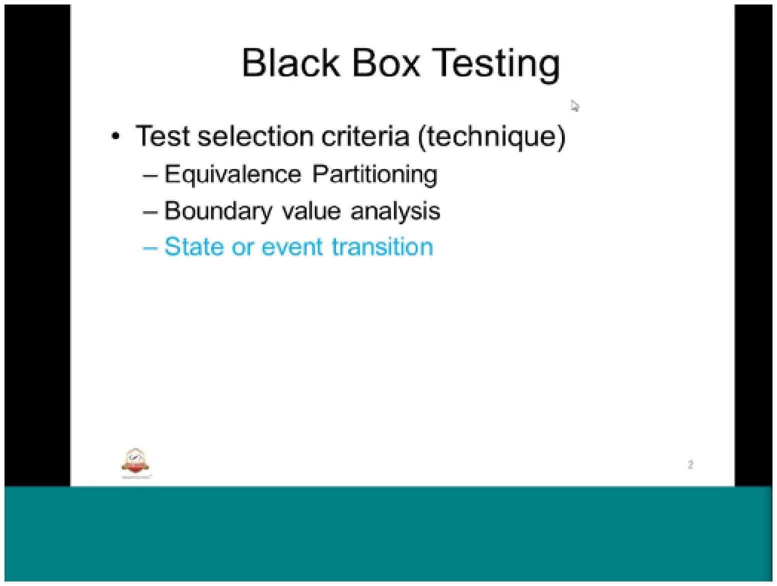
Welcome you to the next section of the embedded software testing the series of any two where we study of testing methods.

(Refer Slide Time: 20)



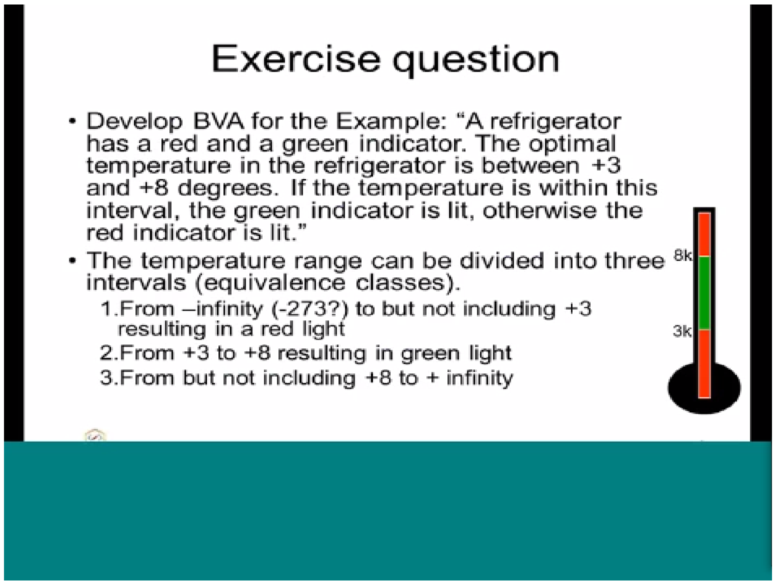
Black Box Testing

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

2

Today we study about state or event transitions we continue on that and we will continue on technique that we have so before that we will see what we have studied in a section 15 we know that given a range something like refrigerator temperature complete to eight we can have all the exercise.

(Refer Slide Time: 00:46)

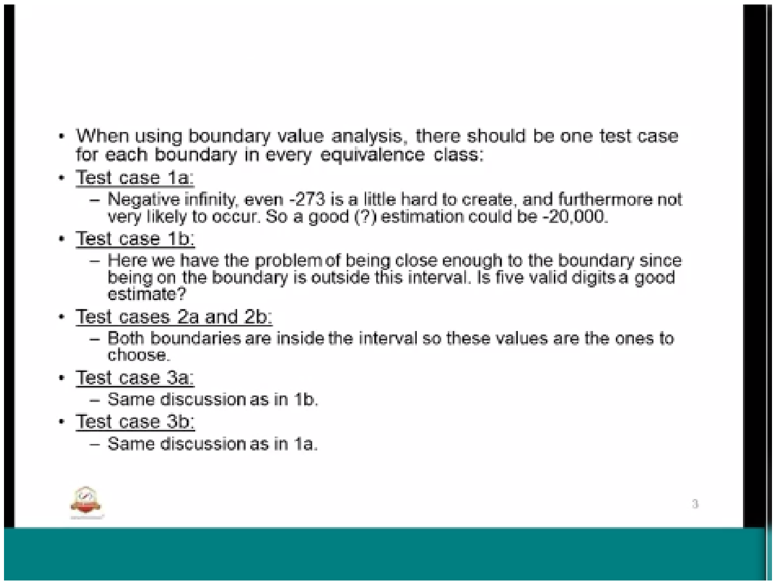


Exercise question

- 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

There are various decisions that we study about the boundary value analysis something with an example of temperature indicator, indicator red link between three and eight with different test case process.

(Refer Slide Time: 01:18)



- When using boundary value analysis, there should be one test case for each boundary in every equivalence class:
- Test case 1a:
  - Negative infinity, even -273 is a little hard to create, and furthermore not very likely to occur. So a good (?) estimation could be -20,000.
- Test case 1b:
  - Here we have the problem of being close enough to the boundary since being on the boundary is outside this interval. Is five valid digits a good estimate?
- Test cases 2a and 2b:
  - Both boundaries are inside the interval so these values are the ones to choose.
- Test case 3a:
  - Same discussion as in 1b.
- Test case 3b:
  - Same discussion as in 1a.

3

We have test case process power that the temperature range about 18 care of negative positive test below the boundary and above the boundary that there is nominal value also the subject to the requirements whatever we have.

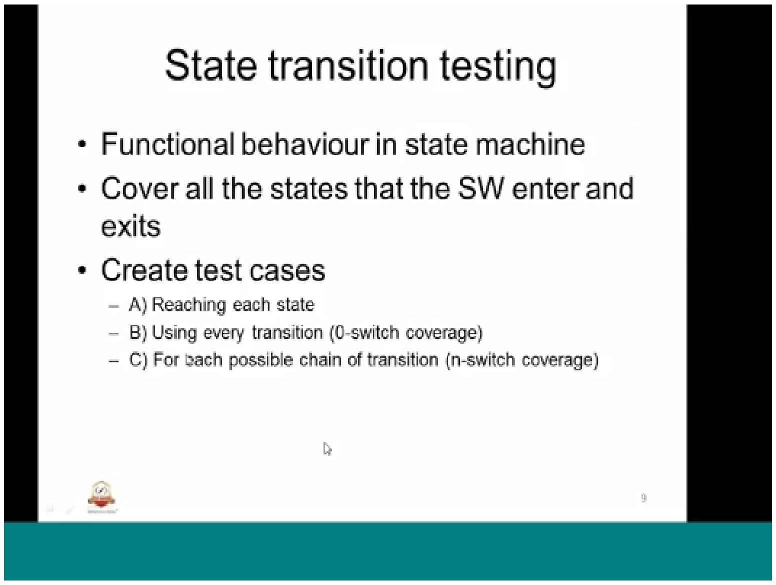
State transitions test we know it is purely based on the change the behavior most in the state based version are state mechanics or implemented in embedded systems that is with the help of the models so today we study about the model based testing questions also before that we will try to understand the what are the state that embedded system can have and how we can test it so we took an example of tap rerecord where it can have state such as off on recording sector.

So three important thing that we reties the memory state equal then where the event based on the event action will be taken care that means if it is off state the tap record is off once if is powered on the power is event the action that is the tap record will be powered off and it will start the initializations and it is a basically a fundamental embedded system nature you take any embedded system we have a power up sequences after power up we have a initialization after the initialization there are the application that will be either initializations or trigger or depending on the type of application, type of embedded system that we have it could launch different applications.

Then we studied about two zones in a, two regions basically which consist between a state transitions testing then, he cooperate the unite of the state the transition can be tested with the help of the behavior and the behavior can be classified into three types one is simple behavior where the system allow the system in the exactly same way to a certain input, independent of the system's history. So it will not worry about the back ground of history you can just drive a value or try to the actions based on the input.

So that is what response is for the certain input then there is second type called continuous behavior where we have the current state of the system depends on it is history in system in such way that a is not identify the separate state that means state keeping on changing we will not being it is a continues behavior of the system then we have state based behavior where the system entering into the different state of the distinguished from other system states based on the system history so this are the basically three types.

(Refer Slide Time: 04:23)



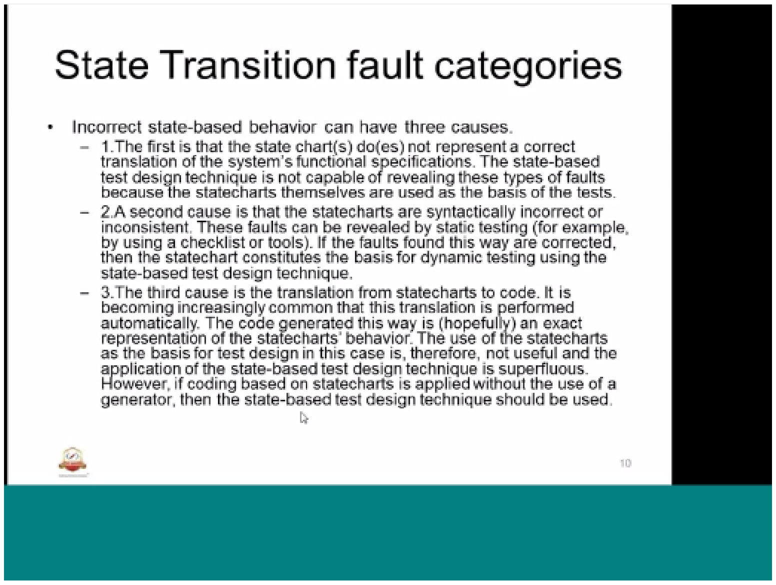
State transition testing

- Functional behaviour in state machine
- Cover all the states that the SW enter and exits
- Create test cases
  - A) Reaching each state
  - B) Using every transition (0-switch coverage)
  - C) For bach possible chain of transition (n-switch coverage)

9

These are all relating to different functional, functional behavior in the system all this functional behavior is under the state machines so we need to cover all the behavior while testing the state ambitions all the entry and exist of each functions of different state it will be revolving within, between the state all this have to be covered so test cases we need to create reaching a inch state and seeing this zero switch coverage in terms of transitions then we have a chain of transitions like n to n switching etc.

(Refer Slide Time: 05:11)



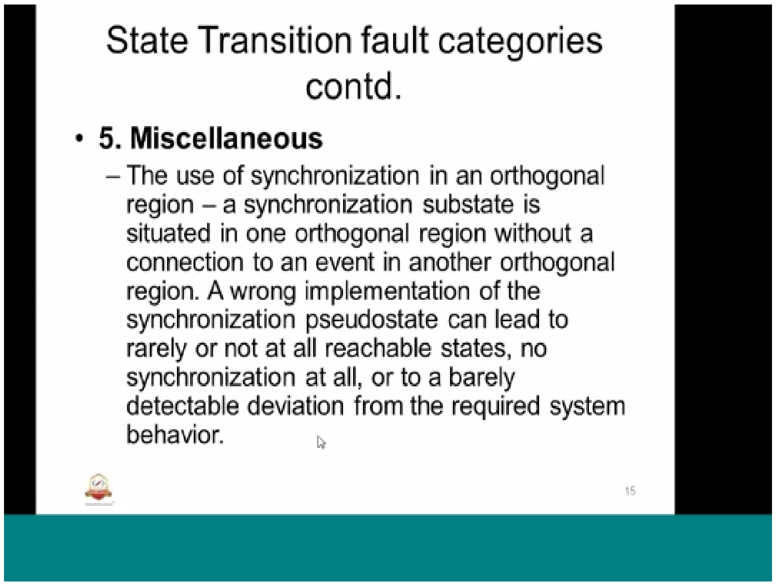
State Transition fault categories

- Incorrect state-based behavior can have three causes.
  - 1.The first is that the state chart(s) do(es) not represent a correct translation of the system's functional specifications. The state-based test design technique is not capable of revealing these types of faults because the statecharts themselves are used as the basis of the tests.
  - 2.A second cause is that the statecharts are syntactically incorrect or inconsistent. These faults can be revealed by static testing (for example, by using a checklist or tools). If the faults found this way are corrected, then the statechart constitutes the basis for dynamic testing using the state-based test design technique.
  - 3.The third cause is the translation from statecharts to code. It is becoming increasingly common that this translation is performed automatically. The code generated this way is (hopefully) an exact representation of the statecharts' behavior. The use of the statecharts as the basis for test design in this case is, therefore, not useful and the application of the state-based test design technique is superfluous. However, if coding based on statecharts is applied without the use of a generator, then the state-based test design technique should be used.

10

Then state transitions fault categories also registered the fault categories can occurs in this states about five state false can be there then we have guards then we have transitions about four transitions faults can be there then we have event four type missing events, hidden paths, and a reactions take place but an identified.

(Refer Slide Time: 05:19)



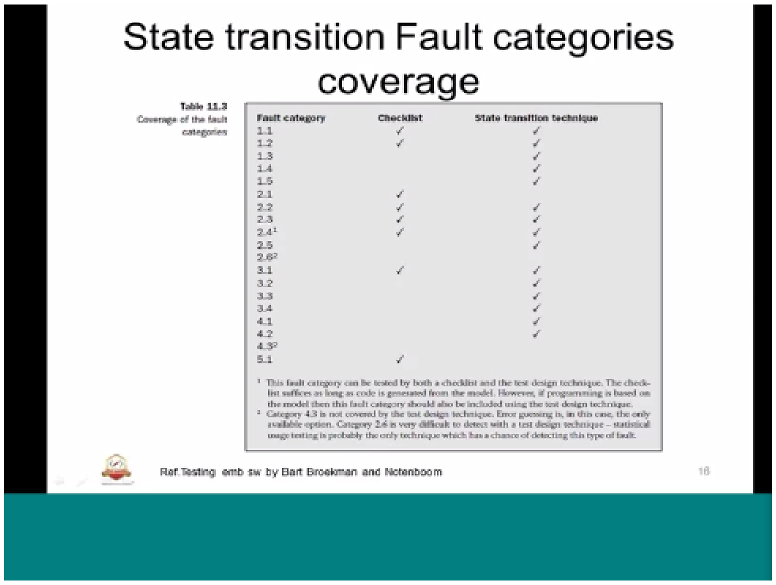
State Transition fault categories contd.

- **5. Miscellaneous**
  - The use of synchronization in an orthogonal region – a synchronization substate is situated in one orthogonal region without a connection to an event in another orthogonal region. A wrong implementation of the synchronization pseudostate can lead to rarely or not at all reachable states, no synchronization at all, or to a barely detectable deviation from the required system behavior.

15

Then we have a synchronization issue we have multiple regions multiple substances are involved and this is connection between these two and state are changing, so that also is the fault or a failure.

(Refer Slide Time: 05:58)



State transition Fault categories coverage

Table 11.3 Coverage of the fault categories

Fault category	Checklist	State transition technique
1.1	✓	✓
1.2	✓	✓
1.3		✓
1.4		✓
1.5		✓
2.1	✓	
2.2	✓	✓
2.3	✓	✓
2.4[1]	✓	✓
2.5		✓
2.6[2]		
3.1	✓	✓
3.2		✓
3.3		✓
3.4		✓
4.1		✓
4.2		✓
4.3[2]		
5.1	✓	

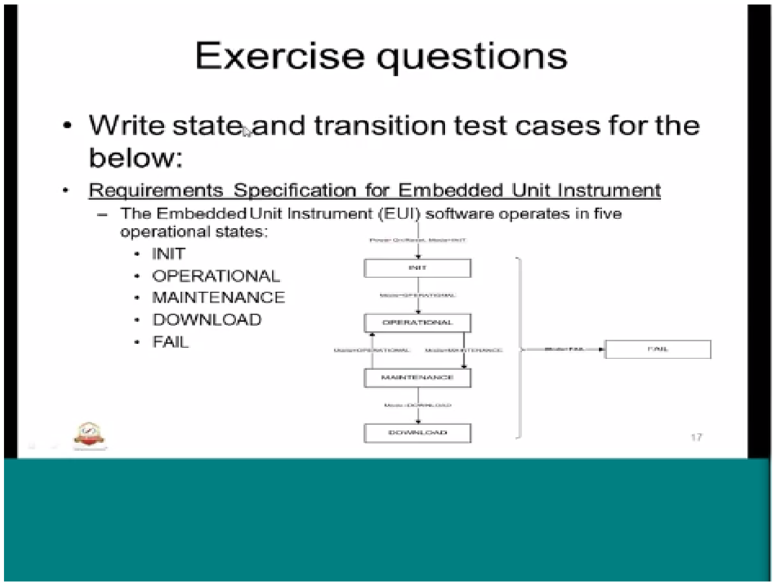
[1] This fault category can be tested by both a checklist and the test design technique. The checklist suffices as long as code is generated from the model. However, if programming is based on the model then this fault category should also be included using the test design technique.
[2] Category 4.3 is not covered by the test design technique. Error guessing is, in this case, the only available option. Category 2.6 is very difficult to detect with a test design technique – statistical usage testing is probably the only technique which has a chance of detecting this type of fault.

Ref.Testing emb sw by Bart Broekman and Notenboom

16

Then we listed out the fault categories in the table and for coverage purpose w use this table to highlight whether that fault have be covered or not so and we also take the state on the syntax link sometimes may not be possible to do abominations or testing technique I just said you in one of the class that those could be tested implicitly with the help of analysis or a structure so that the way the categories coverage we have listed out the check list and state list and we sate analysis techniques and some of the techniques we may have manually or by the instructions or a some other techniques so that why these are not checked here.

(Refer Slide Time: 06:58)



Exercise questions

- Write state and transition test cases for the below:
- Requirements Specification for Embedded Unit Instrument
  - The Embedded Unit Instrument (EUI) software operates in five operational states:
    - INIT
    - OPERATIONAL
    - MAINTENANCE
    - DOWNLOAD
    - FAIL

Power On/Reset, Mode=INIT
INIT
Mode=OPERATIONAL
OPERATIONAL
Mode=OPERATIONAL
Mode=MAINTENANCE
MAINTENANCE
Mode=DOWNLOAD
DOWNLOAD
Mode=FAIL
FAIL

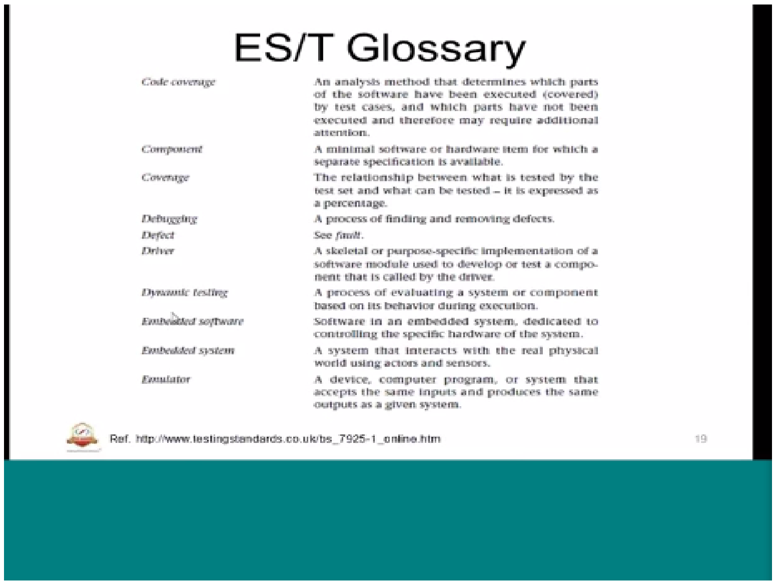
17

So this suppose this to be exercises for the previous sections please make a note of this exercise write the state and transitions test cases for the below okay so I thing in one of the sections you have see till the requirement which I had given, requirement specifications for embedded instrument.

So you can see a different modes I use to take say so now so now you can consider that modes have state so there are about five state INIT, OPERATIONAL, MAINTENCES,DOWNLOAD, FAIL so there is diagram under like this you can use this diagrams and requirement which I have share again probably I will share it.

For that you need to write state and transitions test for the below you can write a simple test cases in defining the inputs and change in transitions of state and write an expected output so that should be the exercises copies sections.

(Refer Slide Time: 08:08)



ES/T Glossary

Code coverage	An analysis method that determines which parts of the software have been executed (covered) by test cases, and which parts have not been executed and therefore may require additional attention.
Component	A minimal software or hardware item for which a separate specification is available.
Coverage	The relationship between what is tested by the test set and what can be tested – it is expressed as a percentage.
Debugging	A process of finding and removing defects.
Defect	See *fault*.
Driver	A skeletal or purpose-specific implementation of a software module used to develop or test a component that is called by the driver.
Dynamic testing	A process of evaluating a system or component based on its behavior during execution.
Embedded software	Software in an embedded system, dedicated to controlling the specific hardware of the system.
Embedded system	A system that interacts with the real physical world using actors and sensors.
Emulator	A device, computer program, or system that accepts the same inputs and produces the same outputs as a given system.

Ref. http://www.testingstandards.co.uk/bs_7925-1_online.htm

19

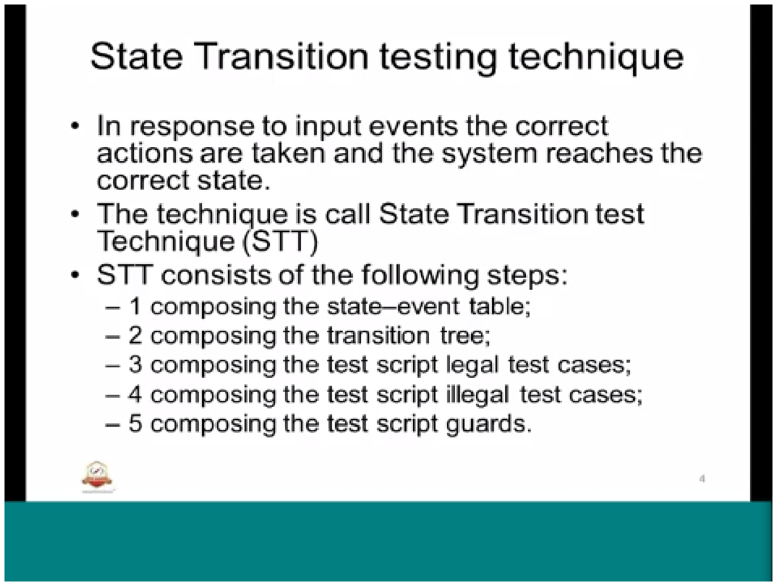
I guess so some of the embedded system glossary we need to touch few how I highlighted here in terms of code coverage we know that code coverage how much code we have tested vs, the complete code so this is nothing but the coverage hop much we have been covered or executed so next one is about the components a minimum software or a hardware item for reaches separate specifications is allowed it means a unit it could be ECU or in aerospace it is called as CSI, CSU whatever it is CSCI etc.

And there is term called coverage the relations between the what is tested by the a set and the what can be tested percentage usually in terms of percentage is the coverage is spoken debugging all we know in terms of finding and removing the reaches we will us this typically a developer set up that can be used for a tester also defect in terms of fault drive a new term a skeletal or a purpose a specific implementations of a software model use to develop or tester a component that is called by the driver but means it is test driver it is called so where we provided the inputs and which will trigger.

Conditions which can be used provided the expected output also that is why we us the driver also because of the test driver then you have dynamic testing a processer of evaluating a system based on behavior during the executions which we know that we study, embedded software is base thing that everybody knows embedded system, embedded software talks about that system having a control functionality.

And all that without the hardware embedded system has both the hardware and the software because software need to work with the hardware then we know emulator qualities so it accept the inputs of the system and produces the output by the system so that is the go to the continuation of the last class so we will see the state or event transition as I said this exercise we need to do by developing the state and transitions cases for the below requirement okay.

(Refer Slide Time: 10:58)



State Transition testing technique

- In response to input events the correct actions are taken and the system reaches the correct state.
- The technique is call State Transition test Technique (STT)
- STT consists of the following steps:
  - 1 composing the state–event table;
  - 2 composing the transition tree;
  - 3 composing the test script legal test cases;
  - 4 composing the test script illegal test cases;
  - 5 composing the test script guards.

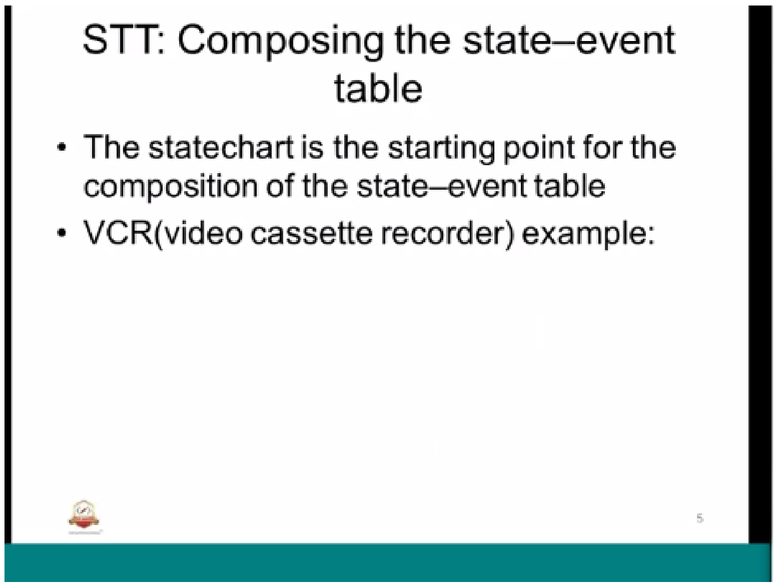
4

So state transitions testing technique we will study so in response to input events the correct action are taken and the system reaches the correct state so that we need to ensure that is what is about state transitions we need to do it that means to make sure that testing we, we will be providing the sufficient inputs so the correct transitions are taken for those inputs and the system is expected for the correct state the technique is called state transitions test technique STT, as I spoken by a book I referred.

In the one of the class the SST be consist of the following steps, there are five steps that state transitions test had composing the state or event table that mean we need to compose the state event table or then composing the transition tree having all the transitions having all the state events table is forward and third one is composing the test script legal test cases so we are coming with the legal test cases that means the test cases which will drive all the transition path then fourth test script is illegal test cases which will take care of the negative cases something like test cases which are not going to happen really in the normal embedded system executions and fifth one is the test scripts guards as we know identifying the guards and scripts how we can do okay.

So we need to know about the state diagram also so we will study about the state diagram in detail that we see the examples so before that we will list the identify the STT, STT you know that state transitions techniques okay.

(Refer Slide Time: 13:24)



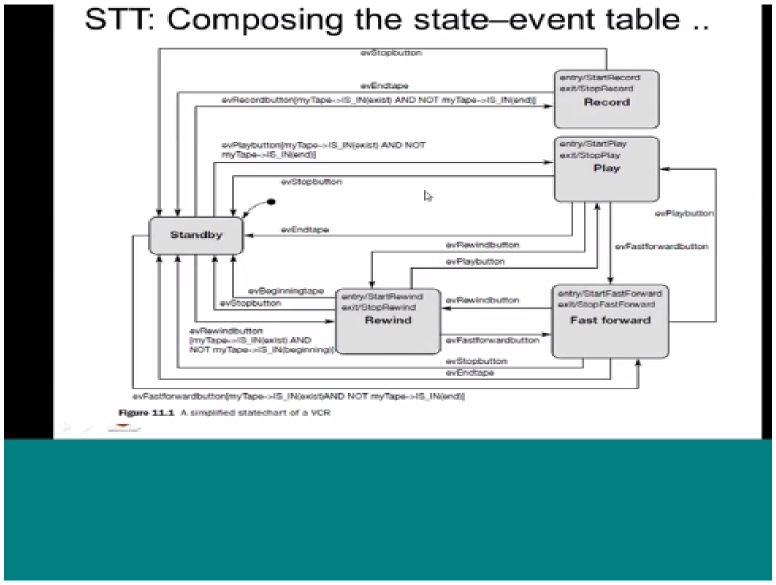
STT: Composing the state–event table

- The statechart is the starting point for the composition of the state–event table
- VCR(video cassette recorder) example:

5

So first step is what we spoke is the composing the state event table okay so state chart is the starting point for the composition of the state event table that means before we draw a state machine table state event table we need to identify the composition in terms of how the state chart is there so we will taken a examples of a VCR.

(Refer Slide Time: 14:01)

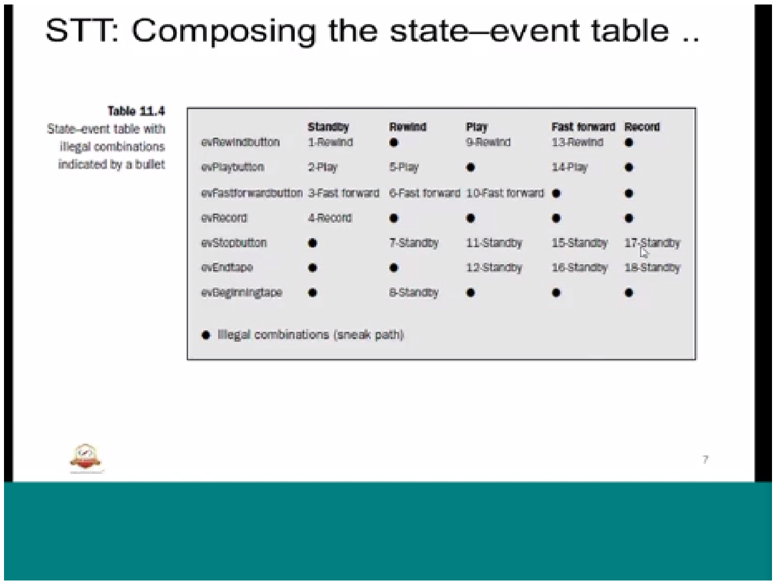


STT: Composing the state–event table ..

Figure 11.1 A simplified statechart of a VCR

So the state is the starting point for identifying this is examples applications so we need to see the composition of the state event table basically it revenues on the, this state first and then events you can see the arrow marks are revenues and the state in the rectangular box so if sate event combinations is mingle then the resulting state of this combination will be incorporated in the table that is we are going to have test cases for legal if it is state event combinations is legal in additions the transitions will be assigned in a number so there is a unique number for that particular , that is identified in the table.

(Refer Slide Time: 14:59)



STT: Composing the state–event table ..

Table 11.4
State–event table with illegal combinations indicated by a bullet

	Standby	Rewind	Play	Fast forward	Record
evRewindbutton	1-Rewind	●	9-Rewind	13-Rewind	●
evPlaybutton	2-Play	5-Play	●	14-Play	●
evFastforwardbutton	3-Fast forward	6-Fast forward	10-Fast forward	●	●
evRecord	4-Record	●	●	●	●
evStopbutton	●	7-Standby	11-Standby	15-Standby	17-Standby
evEndtape	●	●	12-Standby	16-Standby	18-Standby
evBeginningtape	●	8-Standby	●	●	●

● Illegal combinations (sneak path)

7

This series table we will study this first Colum of this table how is the initial space the other columns are consist of a state that can be reached directly from the initial state one transition away, so initial state what are the different state it can enter so that also need to enter next comes the state that a are two state away from the initial state that mean from initial state next to next state this also are there in the third state we can say one examples so until all the state represented in this table.

We are going to continue on the righten side so state event combination are transited to itself resulting in state is accepted must also include in this state that means it can come back also to the initial to see it is going for the standard state that is the one of the initial state we can say there also needs to be listed so using the state char conspiring the video caste recorded that is there in the previous the state event table is drawn as per this, okay you can see the rectangle boxes having the different state we have about 1, 2, 3, 4, 5 states okay so they are about the five state you can see so you need to identify those state first then identification then list out all the lines that are responsible for going to the state.

But that is compile during the starting point of the state which will go into a standby up on power up so in standby state it will continue till we have how many events are there to go to play see one events is there that need to record one more event is there some any other event is there rewind yes but here should be standby and also more last one there is last one that is about fast forward so four state it can go where we need is standby that is standby the one is on play and another is record and another one is rewind another one is this one okay this is, the moving forward method similarly whose is going to play let say standby upon power up we are going to standby next we are going to play from play we can come back standby. Similarly from play we should be able to go to the rewind we know that the tap recorder are easier we can do play rewind and fast forward those button can be pressed so it should that actions for that event similarly the directly from the play we can go to standby if the tap is ended so this is the another connections previously.

They have put all the combinations that VCR can play so this is how this will be denuded basically and taking up of more events let say we are moving by standby to record so there is a guard my tap is in exist and not my tap is in end that means my tap is there without the tap it will not recorded first of all that conditions is a guard first it has to be satisfied then there is event of record button so this to primary input that they are satisfied then it will strict to that state the record state is keep on recording till the stop button is pressed.

So only way to go to the standby is that is the stop button of course there is tap end because this is the one of the guard is false so it will go by the standby okay taking up another sate let consider about the state with in other state let not take the same state and give let see how play and how fast forward can be inter change between them, so you can see state from play to record because from play to record the VSR has to be stopped must then you can go for the record similarly between the playing the fast forward you have link called fast forward button is pressed in the pay conditions so it will go fast forward and if once the fat forward button is pressed again.

So this is what the continues behavior what we have seen for all this events we going to going to draw a table you can see a nice table written here in the table 11.4 of the book state event table legal combinations instead all legal combinations that is indicated by a polite and any other thing that we want to go through here okay this are basically the state events that use to be studied once we have the state chart seen you should be able to draw the table we can see on standby rewind button can be pressed so it will numbered as one play can be pressed two fat forward can be done three fourth is record in similarly.

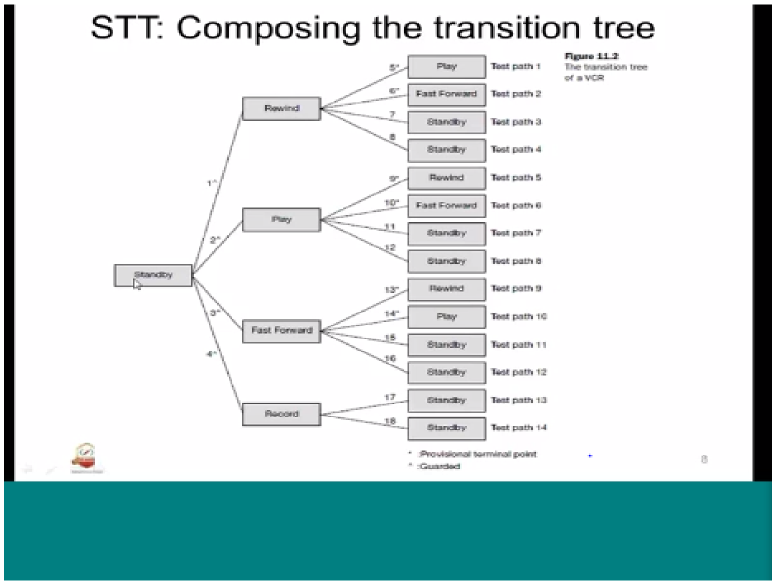
We have standby by two rewind as a fifth and the six one is fast forward button seventh one is standby, standby can go similarly for beginning okay eight has a standby rewinds is to standby you can that is standby so this are, that columns are different state record play fast forward rewind and standby similarly this five columns are sets and this 1, 2, 3, 4, 5, ,6, 7 seven action are there are seven events are there for this event what is state of this state’s okay I will try to explain you 2, 3, 4, 5, these are different state and standby events so event button play button fast forward record stop in tap beginning of the tap this are all event for this events so what is the stand of each tap.

How it is going to take action okay there are bullets of course that bullets are nothing but the legal combinations so we spoke about legal and illegal are what you have called it negative or positive whatever the way you want you want to termite it ever so the legal combinations are something like which would not have any action thou event is valid are illegal event the actions will ended with illegal that means no actions is taken place or it should not affect event for examples when the rewind is happening rewind state with that there is no pointing rewind button up right so we cannot press the rewind or you may press it, it will not take, it will not do action for that that is the legal part of course we need to test illegal also that we need to highlight it or we need to derive table event table for that also.

And when the rewind is happening you cannot be a record similarly like this there are number of illegal combinations standby we cannot do a stop work because it is already stopped and end tap will not happen in the stand by because it is neither rewinding or playing beginning of tap also will not happen similarly while recording is happening rewind cannot happen play cannot happen fast forward cannot happen that record button itself cannot be pressed they is no action from the record button if it is pressed of course it can go for a standby and beginning of the tap it cannot go that is the illegal combinations and you can see there are two standby 17 has standby and 18 has standby so this is based on the events.

So this 17and 18 is different because while record state is happening either gas to press the spot button it is result in a standby similarly the record state is happening it is either has to press the end tap or sorry the user will nit press sorry that tap will end so it also entering the standby similarly fast forward will result in stopping the button it will go for a standby the tap is in that so likewise we need to identify all the events in the state diagram based on the state chart so it is very important to understand the states and different events all this events have to be listed out here so you can see the number of arrows these are entering the event so there are about 18events that are possible with this state diagrams okay so that is the second step so what we spoke about is composing the state event table then we will be how assume the state event with illegal; combinations.

(Refer Slide Time: 27:23)



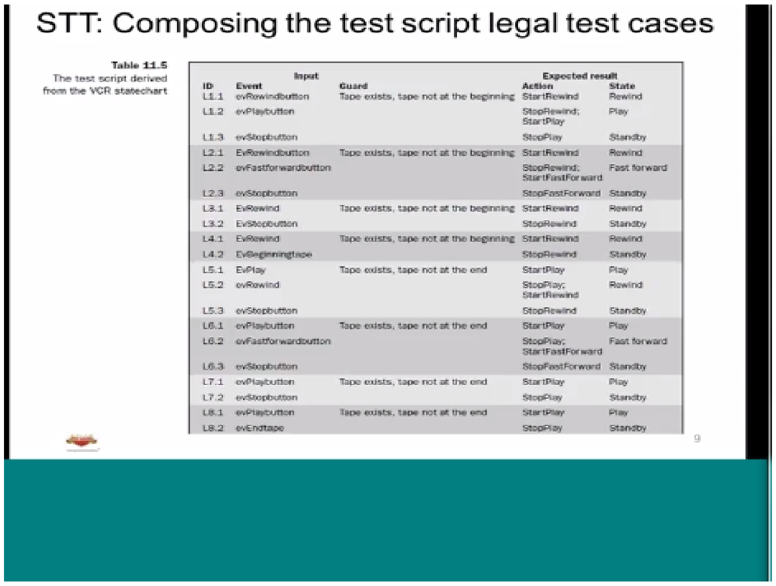
STT: Composing the transition tree

The next one is transitions tree so this is very important aspects also how the transitions are going to happen so why this is important is for a tester we needs to know whether he has covered all the parts different because this fast forward can happen from standby to play to fast forward or the fast forward can happen during standby during rewind like this path we should executed because it is not just a in half the standby the rewind for rewind to fast forward we need to cover this path standby to rewind to fast forward that is why we need just to in half to have the table the next step after the table is listed we need to have transition tree it is nice picture where you can see the first initial point are the initial study that is basic for driving the transition tree because all the parts are going to start from the standby or for the parts that is on the right, right most side is going to take care with the help of elect and four state so that is how we are going to listed out the transition tree okay.

This tree is gone basically by understanding the different events and states that event will trigger so from standby to we know that there are four states that can enter rewind play fast forward and record each sub state of standby that is rewind play fast forward and record can enter into six sub states it can go for a play it go can go for a fast forward it can go for standby so this path will need to executed did how many parts are there now likewise we need to count all the parts the first part is this standby two rewind second is standby to play so there are four parts are there once we have this four part with, rewind with play.

So this the test path one so equal to test path one we need to have path one and path five done because standby to rewind to play we need to have right similarly seventh path eight path like this continues test path are different so each path are legal combination basically we can see above 14 parts are there so which will cover the completely testing of this events so the note got it so the standby to rewind is with the help of the conditions that we spoke about with that guard is there when that straightly enter so other are possible terminal point that mean you did not to go behind them so you see that the fast forward is last terminal point in that places also terminal standby is not a terminal because standby got a different path so likewise we need to have a conditional part defined okay.

(Refer Slide Time: 31:22)



STT: Composing the test script legal test cases

Table 11.5
The test script derived from the VCR statechart

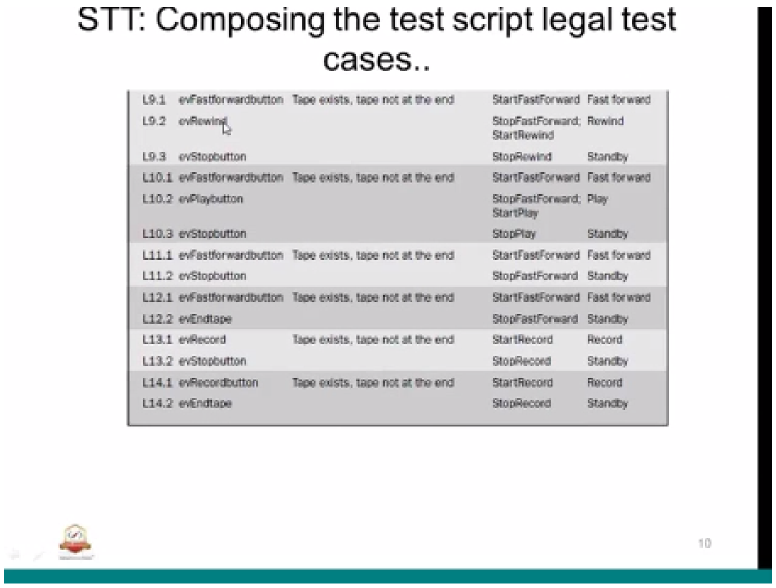
Input			Expected result	
ID	**Event**	**Guard**	**Action**	**State**
L1.1	evRewindbutton	Tape exists, tape not at the beginning	StartRewind	Rewind
L1.2	evPlaybutton		StopRewind; StartPlay	Play
L1.3	evStopbutton		StopPlay	Standby
L2.1	EvRewindbutton	Tape exists, tape not at the beginning	StartRewind	Rewind
L2.2	evFastforwardbutton		StopRewind; StartFastForward	Fast forward
L2.3	evStopbutton		StopFastForward	Standby
L3.1	EvRewind	Tape exists, tape not at the beginning	StartRewind	Rewind
L3.2	EvStopbutton		StopRewind	Standby
L4.1	EvRewind	Tape exists, tape not at the beginning	StartRewind	Rewind
L4.2	EvBeginningtape		StopRewind	Standby
L5.1	EvPlay	Tape exists, tape not at the end	StartPlay	Play
L5.2	evRewind		StopPlay; StartRewind	Rewind
L5.3	evStopbutton		StopRewind	Standby
L6.1	evPlaybutton	Tape exists, tape not at the end	StartPlay	Play
L6.2	evFastforwardbutton		StopPlay; StartFastForward	Fast forward
L6.3	evStopbutton		StopFastForward	Standby
L7.1	evPlaybutton	Tape exists, tape not at the end	StartPlay	Play
L7.2	evStopbutton		StopPlay	Standby
L8.1	evPlaybutton	Tape exists, tape not at the end	StartPlay	Play
L8.2	evEndtape		StopPlay	Standby

9

The next after the composing the transition table sorry the transition tree we need to have the legal test cases right so test script we are going to derive for this legal test cases this is based on the then the guard that means the guard also need to be taking care in terms exist the tap not at the beginning it is in the middle tap exist it is not in beginning it is the one of the guard tap exist tap not at the beginning so all this are key inputs for the event to happen and for this individuals test are the legal scripts we have action.

So what do you mean by actions the expected action is start rewind the state where it will enter rewind state stop rewind as the result of that it will go to the next as the play and the resulting state is play so in second test cases play button is pressed what it will happen if it do as rewind state it will stop and the start the play it will go state play state similarly stop button is pressed the play because in the previous state we stopped we will go to the standby when we will have the rewind button so likewise we are going to restart all the legal scripts we can have any combinations of this but there should be based on this path and the legal events or legal combinations that we have similarly we are going to have legal combinations functions.

(Refer Slide Time: 33:28)



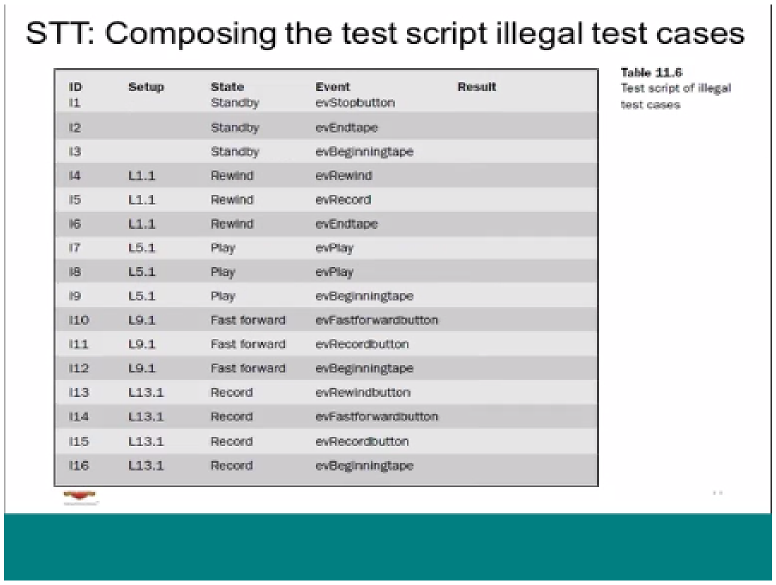
STT: Composing the test script legal test cases..

L9.1	evFastforwardbutton	Tape exists, tape not at the end	StartFastForward	Fast forward
L9.2	evRewind		StopFastForward; StartRewind	Rewind
L9.3	evStopbutton		StopRewind	Standby
L10.1	evFastforwardbutton	Tape exists, tape not at the end	StartFastForward	Fast forward
L10.2	evPlaybutton		StopFastForward; StartPlay	Play
L10.3	evStopbutton		StopPlay	Standby
L11.1	evFastforwardbutton	Tape exists, tape not at the end	StartFastForward	Fast forward
L11.2	evStopbutton		StopFastForward	Standby
L12.1	evFastforwardbutton	Tape exists, tape not at the end	StartFastForward	Fast forward
L12.2	evEndtape		StopFastForward	Standby
L13.1	evRecord	Tape exists, tape not at the end	StartRecord	Record
L13.2	evStopbutton		StopRecord	Standby
L14.1	evRecordbutton	Tape exists, tape not at the end	StartRecord	Record
L14.2	evEndtape		StopRecord	Standby

10

Is a continuous of legal combinations functions there are about 14 legal combinations are listed again each one is having is own sub events ha you there are test part of 14 parts are there so each 14 part can have another entering and existing so this have to be existed with the help of legal test cases next one is the illegal test cases so illegal test cases we know that are those when the rewind is happening there is no pointing press in the rewind similarly when the recorded is happening this five the bullet items the rewind cannot be done play button cannot be done and fast forward cannot be pressed and recorded cannot be done.

(Refer Slide Time: 34:32)



STT: Composing the test script illegal test cases

ID	Setup	State	Event	Result
I1		Standby	evStopbutton	
I2		Standby	evEndtape	
I3		Standby	evBeginningtape	
I4	L1.1	Rewind	evRewind	
I5	L1.1	Rewind	evRecord	
I6	L1.1	Rewind	evEndtape	
I7	L5.1	Play	evPlay	
I8	L5.1	Play	evPlay	
I9	L5.1	Play	evBeginningtape	
I10	L9.1	Fast forward	evFastforwardbutton	
I11	L9.1	Fast forward	evRecordbutton	
I12	L9.1	Fast forward	evBeginningtape	
I13	L13.1	Record	evRewindbutton	
I14	L13.1	Record	evFastforwardbutton	
I15	L13.1	Record	evRecordbutton	
I16	L13.1	Record	evBeginningtape	

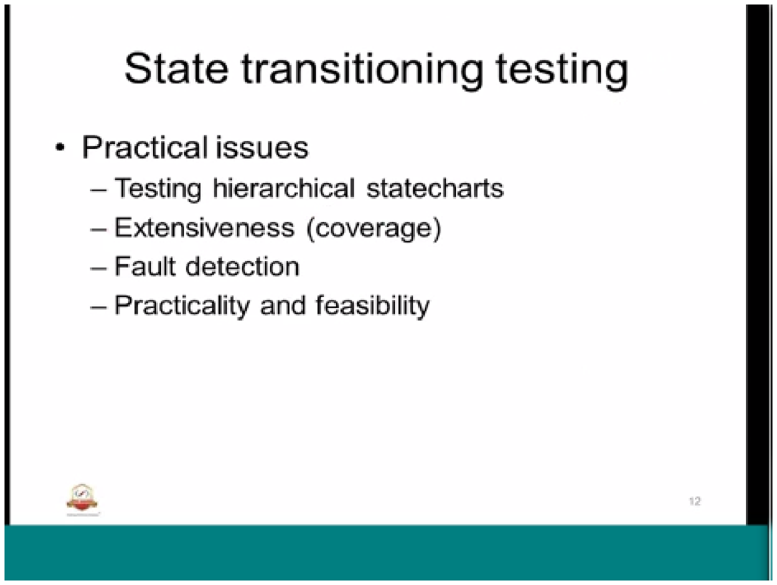
Table 11.6
Test script of illegal test cases

So these are all illegal combinations for each of these illegal combinations we are going to identify uniquely with IDS illegal one illegal two and illegal three and there are about 16 illegal events are the combination that can be done so let see how many of them are here about 3 to 6, 9, 12, 16 illegal combination are there so same thing is explained here so set up L, L.1. so what is that L.1 is what we have seen here the rewind button tap not at the beginning that means this is the card and action is rewind so rewind state is it is there now how to read the rewind state rewind button pressed and when the rewind is happening what is the illegal action we are going have we going to press the rewind that is the illegal action.

And it is first the initial set up we do not need to mention any of the because that is the default state we are going have a stop button or end tap beginning of the tap in order to result in anything so that is the illegal state, test cases of course we need to except so having this illegal we have the complete coverage of the state events so this is the four step that we are going to do so what are the four step.

That we have seen first we are going to have a transition event table then we are going to have a transition tree then we are going to identify the legal test scripts and illegal test scripts state event table in is the one transition tree when the legal scripts and the illegal scripts this four steps are very important for drawing the state event of course the last one is doing well test guards we can do that, it is amounted tree but it is depending on the conditions. Mostly we have covered with the help of these legal and illegal scripts so we can split that also okay.

(Refer Slide Time: 37:03)



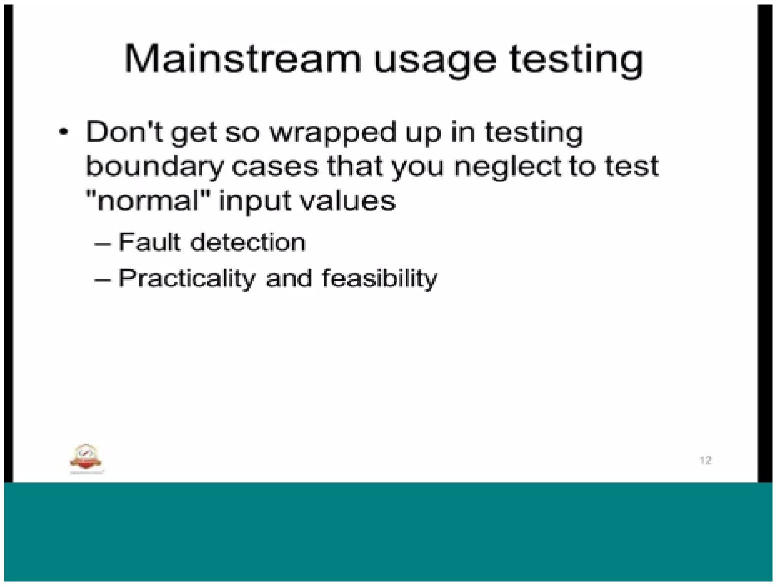
State transitioning testing

- Practical issues
  - Testing hierarchical statecharts
  - Extensiveness (coverage)
  - Fault detection
  - Practicality and feasibility

12

So while doing this state transitions testing so they are sudden changes in terms of practical so what are those we will study about that, so testing hierarchical state charts that means the state chart can have a hierarchical in terms of main systems and sub systems it may little impact to all those combinations.

(Refer Slide Time: 37:59)



Mainstream usage testing

- Don't get so wrapped up in testing boundary cases that you neglect to test "normal" input values
  - Fault detection
  - Practicality and feasibility

12

It could be term or bottom down planner it could be there so it may be difficult to do or achieve the test and hierarchical state chart so if like arrow key if the detail are not much laid out or the detail of underline are not known very well so it is more particle to use the a top line approach in that cases because the details we do not know so we will start with the top down manner because hierarchical we have doing this is what that we need say like we know in which hierarchical we have standby two the next level.

If the complex is terms which has it is own substrates and that play a fast for what and the details for that model or not know so then this difficult to do the bottom of the onwards should better to start with the initial or the top down approach that is what it means so the supper state it is also called supper state so that is what is the matter for us to test it so only the incoming and outgoing turn in that state is very important in that cases to identify to cover as much but there are the bottom of approach the lowest level of hierarchical key is tested but if the detail are very much available successively higher level, higher levels of the integrations are coming tested and finally they completed the tested.

So in complex system what will happen is the top down approaches may not be not be possible conferences because we may not have the system completely and we have a clarity about higher level aggregals so in that cases the bottom of the approach is suitable once we have the integration complete done then higher level hierarchical top down approach can be taken care next one is the extent fewness that minas the coverage as I said we have a different stages we have a legal combinations of the scripts.

That can be developed for the legal state entry and exist and we have also illegal state even combinations all this need to be covered in order to coverage but it is very much essential to have a all this pertain or sub pertain parts in completed systems it will be tricky to have the coverage some ravage it is difficult to cover especially the path when it is straight forward standby to play and play to standby it is easier but from standby to play from play to fast forward and fast forward suppose has other software internally it may be it will tricky or in practical in that case we have to other methods to overcome those this particle issues that what it means so the extension is state based test method can be modified by deciding then what exertion the depending . it depends on the method.

So the transitions can be tested individually in that cases but it is also possible to test a combinations of conditions then it is possible we should be doing that if it is not possible we should be testing it alone that means the each transition can be executed or done individual suppose we may not have provision or suppose we may not have a provision to test this part complete right on standby to play to this so the best alternative is change the create of this try to create of this then try to create of this first executer this then try emulate as much because we know that play are entered try to execute this path try to test that path you start with this so end with this so one executions is gone then you start with is end with this or it could be further next level so individual.

We can cover where there is practical issue we are facing so that is what is the extensiveness there are the coverage aspects if there are challengers so we know that test part coverage is the number of path executed vs the total number of coverage N to 1, switch coverage 0 to N coverage all this coverage percentage that we are going to have it as report next one is the fault detections the state base hidden techniques discussed here this based on the test based level so minimum test effort is required for state base behavior have once if we have this platform ready the entered system is based on this.

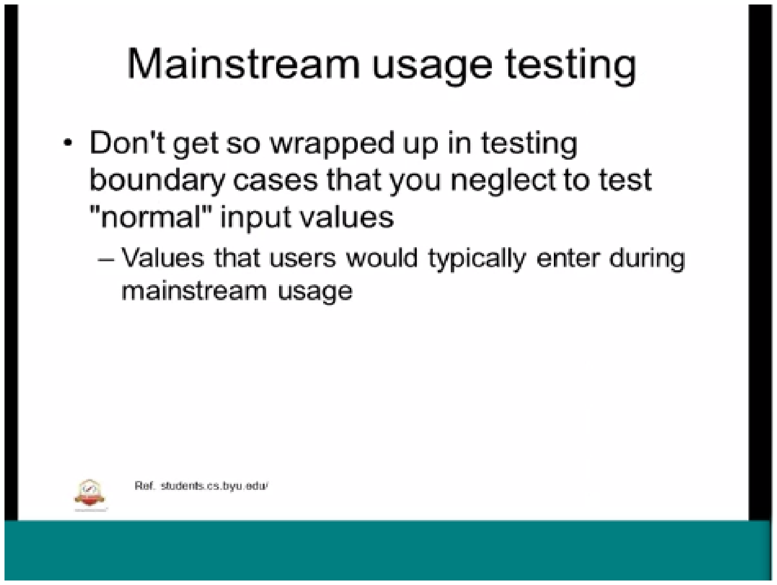
So we do not have this change it again if there are system so we should be able to find the faults easily but sometimes what will happen is we may to change right from the beginning that means through address a fault single fault that could be any of the deepest we may have to exchange the complete part so that is the challenge in terms of detecting the part so fault categories all fault categories may be tricky challenging in terms of testing so in that case the individual the different states can be exercise and find the fault and take it detected for the other states which are exercised in additions to that.

So we need to have a choices which will help us in a isolating the fault based on the label what we have drawn in the slide then it is a disability that means whether before I said the state basic testing we should have a visibility whether I can practically achieve this is very important so that is the one of the practically issues until we test it otherwise we may not be able to plane it appropriately we may have to revise our plan when we really do the testing so the testing of system before I said does not always sector to be simple.

We have basically the behavior of the system is hard to return because the behavior is consoled to extent to large extent from outside the observation it could be a different what we see outside then we start testing embedded system having a object oriented type of implement so where the data encapsulations it may be difficult in test such cases so a event can be two cased in the sense before the system reaches to the state that can be observed so we need to know the inter cases of those that is where the feasibility will be process in terms of understand that and drawing the transition test mechanism sometimes it may be impossible to do it to distinguish between individual transition and spots at the lower level so in that case either we should be doing it the emotions of instructions. So that is the solutions what I have observed where we have a practically so those are the practical issues composing the testing which is very important to have a straight ambinations that is because the end system is mostly do with the straight ambinations but there are practical challenger.

In the in issues for doing that state based testing so we need to work on that with the help of all this best expertise what we can have in terms of system knowledge and the state that can interferer okay there are that is above the state on the center so we will have this same exercise has what they have here straight testing for this embedded unit which we able to do it is an exercise for state transitions testing so what we need to do is we need to have this four steps done so let us try to do state event table versions three a script a legal illegal we have to do it so that is in the exercise courses state transition testing .

(Refer Slide Time: 49:05)



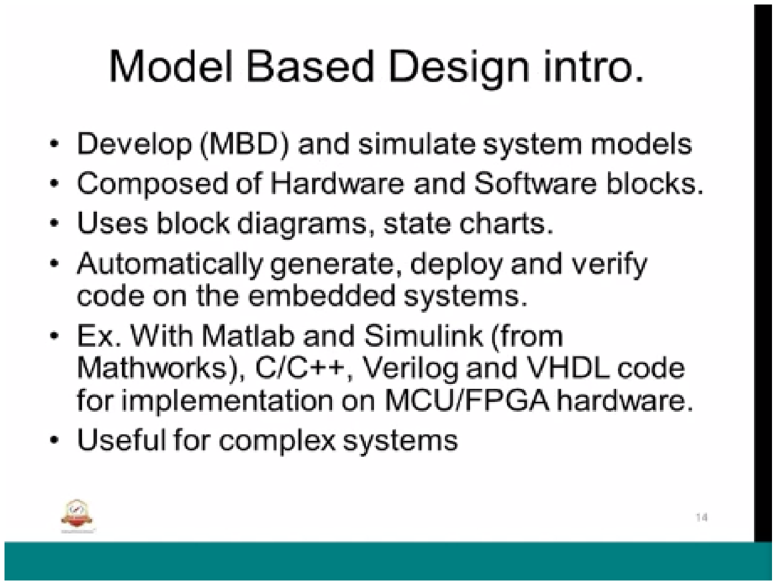
Mainstream usage testing

- Don't get so wrapped up in testing boundary cases that you neglect to test "normal" input values
  - Values that users would typically enter during mainstream usage

Ref. students.cs.byu.edu/

There are other types of test we will study in a brief we just to highlight in terms of what shot of test are availed we will not go in detail main stream usage testing it is called since this is the one of the testing technique the idea is do not get the wrapped up in testing boundary cases that you neglect to test normal input so this will be interesting why because we strictly we go with the theory of all that by doing that we may a good or a nominal inputs itself so we will do all the boundary but this treat for a normal range there is no pointing having need test technique taking great of that so we will to have an understanding and have a good test cases basically so that is what is called mainstream usage that means the nominal usage of the system that we need to have so did you called as mainstream usage testing.

(Refer Slide Time: 50:13)


Model Based Design intro.

- Develop (MBD) and simulate system models
- Composed of Hardware and Software blocks.
- Uses block diagrams, state charts.
- Automatically generate, deploy and verify code on the embedded systems.
- Ex. With Matlab and Simulink (from Mathworks), C/C++, Verilog and VHDL code for implementation on MCU/FPGA hardware.
- Useful for complex systems

14

So we will study about a now model based testing so we will have small introductions section and we will study what are the model based testing and all that then we know that model is a behavior of the system a simple as that so what is model based system so we know the system is designed with the intent of what do you which behavior what is going to give the result what is going to execute so what are the elements that it requires to continue it is operations smoothly as well as under some of the up normal conditions.

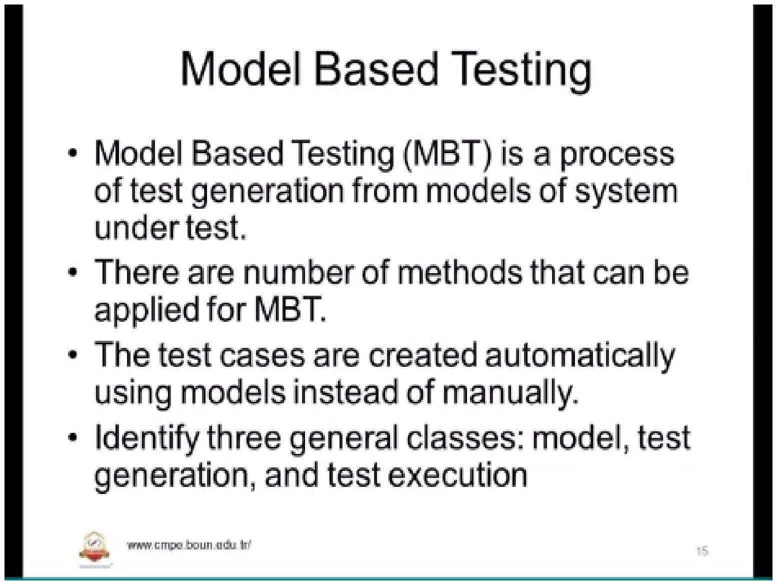
So that is about model so those finite elements of the system are developed in teams of various models that the system is going to have that is what is about the model and model based design is based on that only first we will a model based development terminal is called so we have to develop the models so models are develop with the help of the matlab and simulink so from a mathswork and of course there are object oriented C++supports are also there like such as visual C++ supports sometimes developing the implied models but mostly it is used for developed it above models VHDL are in the examples okay.

So we have a model why because it is useful to analysis the system and it is useful to simulate if there is a problem we have a bigger picture of the enterer system with the help of the models that is why so mostly the embedded system where compliable they go for model based the model will have hardware as well as software block hardware like interfaces such as ualt SPI SCI or a flash ITC all this will be part of the model so all this the part of the hardware and that also can be modeled.

So all those individuals units are called as blocks and those blocks. So for model based design that has block diagrams may be we will have a few samples with the next sections so the advantage with that tools that we have today which helps us to develop the models you have you can generate the code not completely probably so next event it will help the user to develop from the starch you should not develop from the starch so it automatically generate the code which can be deployed and verified.

On the embedded system it means we have a developed a model software blocks and some hardware blocks and with the help of the tool it will generate the at least the frame work so how it interacts and what are the parameters and all that so with the help of the frame work and some fine training with respectively en target which will be able to have the en systems so that is where the advantage of the model based development is base line so the examples is, I might have some link and verilog and VHDL code. So here we have complex systems this model based development is very useful that is why they go for model design and development okay. Ones we have the model based availed or model based design is developed we should able to test how we can test it.

(Refer Slide Time: 54:59)



Model Based Testing

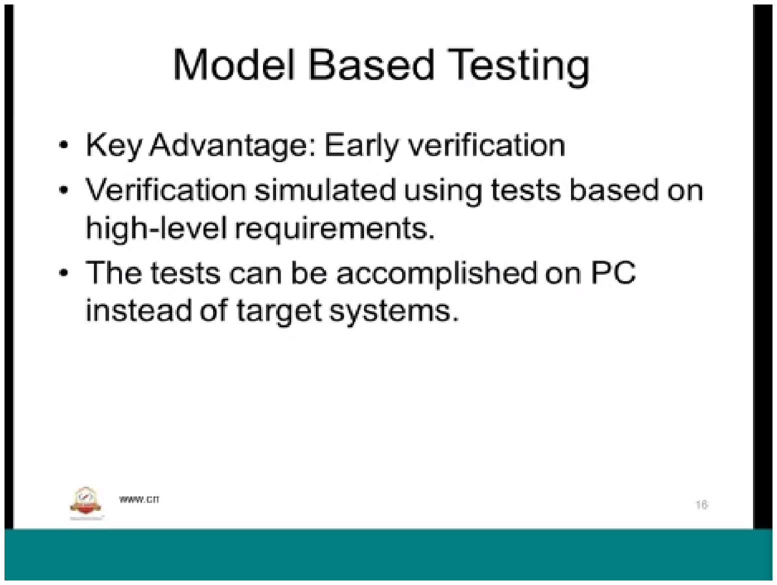
- Model Based Testing (MBT) is a process of test generation from models of system under test.
- There are number of methods that can be applied for MBT.
- The test cases are created automatically using models instead of manually.
- Identify three general classes: model, test generation, and test execution

www.cmpe.boun.edu.tr/

15

So we can also do the testing of model with the help of technique called model based testing where we apply the same terminal at the as model based development here it called model based testing so a model based testing is a process of test generations from model of system under test that means we develop the models which will help in doing the testing in terms of generating the test cases or testing it basically so there are numbers of methods that are given used MBT model based testing we studied few of them in the next class so the test cases are created automatically same as development we have seen generations is there similarly we have test cases generated automatically from the models we should have this manually of course there are disadvantage also for complex system it is useful where we have time and study of the, we have a chance to read that automaticity testes but there are the simple systems where we have various interfaces and through models are available it is better to go for manual black box set up is that what we have studied earlier.

(Refer Slide Time: 56:33)



Model Based Testing

- Key Advantage: Early verification
- Verification simulated using tests based on high-level requirements.
- The tests can be accomplished on PC instead of target systems.

www.cn

16

So that is our advantage and disadvantage so basically broadly three class are used for model based system first is the model itself test generations and test executions how these are all involved in the model based testing so this are the some of the advantages we can have a, we need to identify the model how we can verify at least we have a pre memory understanding of a models in terms rightly modeled are denied.

So it sort of a advantage verifications simulated using test based on high level requirements but this we can way out the verification mechanism with the help of high level requirement only the test can be accomplished on PC instead of target systems we do not need a target systems we need to have a knowledge of the target systems with the help of that we can develop the model based testing on the computer only not on the targets okay

(Refer Slide Time: 57:33)

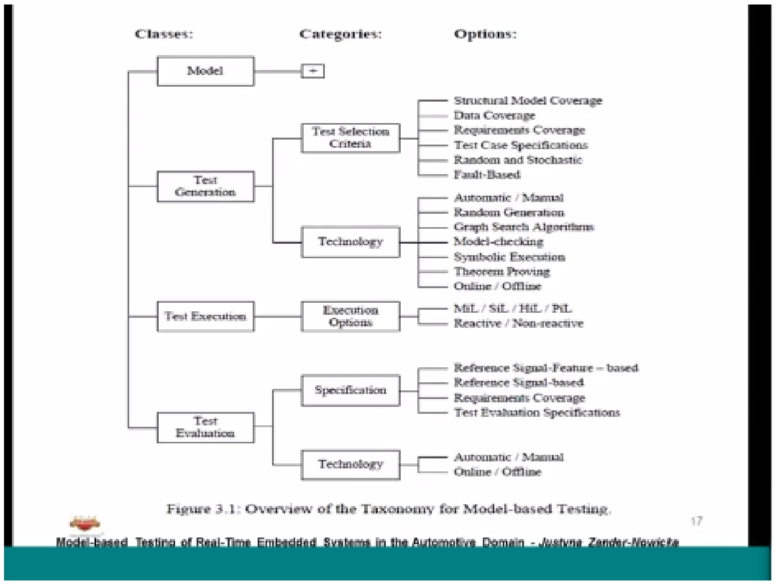


Figure 3.1: Overview of the Taxonomy for Model-based Testing.

So there are other taxonomy overview of the model based systems we mean study that model based system and is inter cases in the next class okay to conclude we will revise it what we have studied and that studies we need to have so four state transitions technique steps we need to apply for the sample embedded unit having five modes of state or operations we need to operational maintainers download we will study model based techniques and how it can be detailed in the next class.