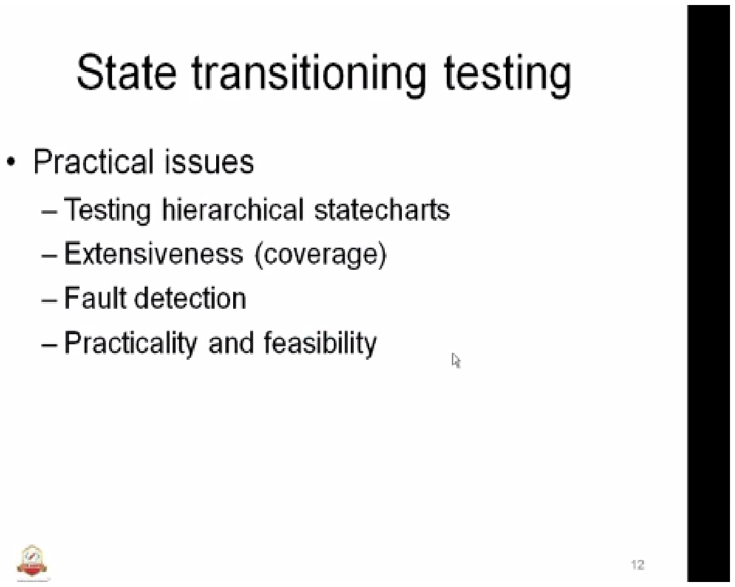
The next session of embedded software testing unit to series where we study about the testing methods today session we will touch on the mo0del based testing and his technique and details before that I will in the last session we had gone through the state or even transitions of testing okay in today session we will continue that because the last session is important in terms of state transitioning and the model based design and development and testing is based on the state, state machines grammars and all that so what we study last time is state transition testing technique in which we studied about composing the state.
And composing transitory tree and test script legal cases, test scrip illegal test cases it is something like robot skills, illegal mean it components analytical are something like that it is beyond the normal test cases or so called illegal test cases and test script guards which is the optional so other thing is we took an example of a cassette recorder or video cassette recorder which has about five state standby, play, record, fast forward, rewind and as per as the practicality concerns we studied about all the transitions guards conditions.
And the events that are going to happen and accordingly we know that for all of this we listed out a state event table which identifies about 18 types of state events and doted ones are cannot be achieve which are illegal but the system should accept the tester try to o this doted ones as a robustness but the system should not accept as the state events that is why this are all doted and that is called as illegal combinations.
And different events are standby sorry state are stand rewind play fast forward and record also we had drawn a tree which is talk about the path it is nothing but the control path, control path that is something called the data path which we do not need to control coupling, data coupling very important formula it is been used in the embedded system the formal purpose on control coupling control path which is nothing but the executions of the program taking or considering the different events that are happening in the test events.
So that is nothing but the transitions tree so we have an about 14 test path starting from standby. And it is sub state is rewind, it is sub state is play, fast forward and then standby like this play also have so all this combinations we are seen the different part that are taken up in the state event transition tree where we have the primary event of the standby then we the rewind play fast forward record etc so all this path have to be tested then we have to on a table or the state chart table for the VCR have a identifier with identification legal 1.111.3,2.223 etc all this are basically event based.
So there is a input and the condition next one and the expecting result so this is how the events to be done in terms of state transition testing continuous on that so we have a illegal test case also what we have seen in the doted ones having identifiers the I-1, I-2, I-3, till I-16 so set for this similar to the mingle one to the , where we need L1.1 set up that means the rewind the state is required and rewind to event rewind we need to know which is the illegal scrip basically. Similarly all this have been listed out.
(Refer Slide Time: 05:35)



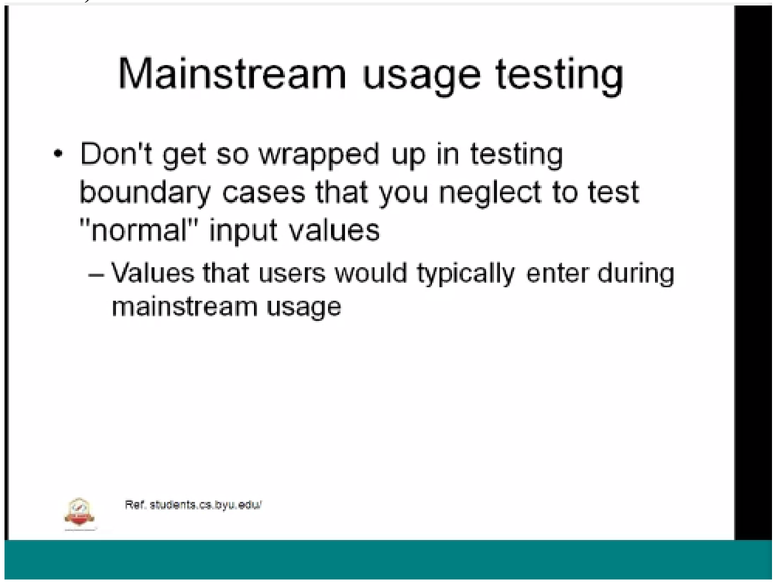
State transitioning testing

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

12

So there are practical issue which we have seen hierarchical state charts to draw it may be little challenging we need to have a practical knowledge or system knowledge requirement of course will help as a then coverage the extensiveness we need to have it so state charts are very important in doing this then defecting the fault practicality and feasibility in achieving all this so this are the some of the practical issues in the state transitions testing.

(Refer Slide Time: 06:05)



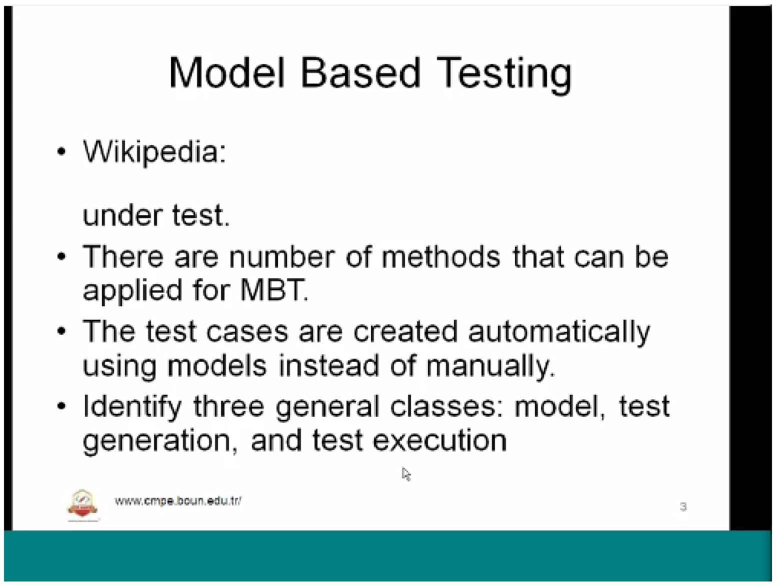
Mainstream usage testing

- 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/

So that is do be handed there is a another term called mainstream usage testing it talks about we should not struck our self, our self into boundary case only we also see some of the good values as well in terms of normal as well as the normal expected output so typically that will be used when the system is run that all we will do it okay. So model based testing and designing today I will go back to that slide okay.

(Refer Slide Time: 06:47)



Model Based Testing

- Wikipedia:

  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/

3

So before the model based testing the model I have been developed in the development face with the help of model based design and development such a matlab or simulink there are of course different tools also where we have VHDL etc those are specified for the hardware for complex system we use model based development and testing basically it uses different models for this system under the development test.

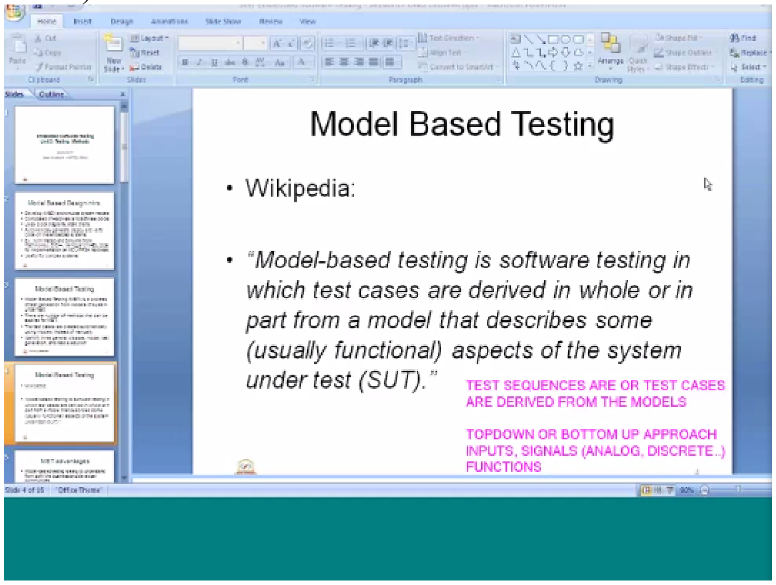
Those models are nothing but different diagrams chart so those will be done ones we have the those models and those models with the help of tool cab be used to generate the code and that code can be deployed into the target the basically that code can be generated in host and can be deployed into the target and the target can be verified with the help of the target based executions but before that we may have to do some work that is not hundred percent what is this called directly use able with the help of auto code generate code, code.

So for that we need to little modify the auto generated code similarly for verifying it we do the same modeling concept for test that means basically is model is a frame work for us to go for the testing which is nothing but the model based testing we will study about it before that I just want to describe in brief about the model based design so similarly there are lot of tools concept wise we have UML unified modeling languages.

Then we a SML that is system modeling language so these are basically dealing with transitions diagrams state chart all this type of model called representing the system in terms how it is executes and how it is getting modeled so that is how these are very useful once we do the model they did not generate the code and with the help of the target they will add additional code In terms of wrapping the generated code into the target then they executed same philosophy.

We use for testing also same model concept the used so that is how it is called as model based testing so let us see about the model based testing in the next line okay so model based testing with the process of test generations from the model as I explained in the previous slide for this system under test there are number of methods that can be applied for MBT, that is model based testing the test case are created automatically using the model instead of manuals that means we

have created the models and the test are generated automatically from it is models okay test cases are created automatically to saving test case with the help of test hook we will generate the scrip.
And then execute of the target board, target board is already having the modeled code which is the code executing on the target so basically three classes are getting identified model, test generations and test executions instead of basically the class they say about MBT model based testing.
(Refer Slide Time: 11:07)



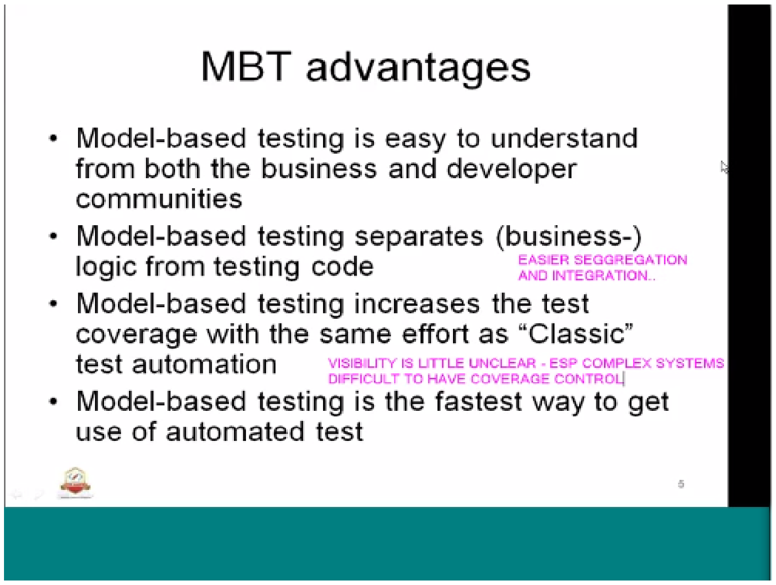
Model Based Testing

- Wikipedia:
- *"Model-based testing is software testing in which test cases are derived in whole or in part from a model that describes some (usually functional) aspects of the system under test (SUT)."*

TEST SEQUENCES ARE OR TEST CASES ARE DERIVED FROM THE MODELS

TOPDOWN OR BOTTOM UP APPROACH
INPUTS, SIGNALS (ANALOG, DISCRETE..)
FUNCTIONS

So definitions let we see what is model based testing going by the Wikipedia all thou they are several, most of them carry the same basic idea about and model describing the system in the system of some sort and some way of using that model to get the test case or to generate the test cases so basically we called as test sequences, the sequences or test cases are derived on the model so that is the fundamental about this so the model based system is software testing.
In which test case are derived in whole or part gat can be completely for sub model or smallest of the model or the enterer model where we have the top zone a I said in one of my class or bottom any of the approach where we considered the smallest of the units and go towers upper side something like the interrogations approach as a bottom of we can use otherwise top down where we have high level requirements that means is very clear.
And we have the models and we can use the top down approach in terms of whole stick testing so model is the software testing in which the test cases are derived in whole or in part forming model that discuss some function aspect and, that means the model basically defines and describe the function aspects and each models will have it is own functionality features inputs outputs what are the signals could be analog describe whatever so all this will be part of the functions which are basically use to describe the models.
So all this useful in terms of developing the test that can whole as well as the unit so that is the definition of model based testing okay so let see some of the advantages of model based testing it is very useful in complex system.
(Refer Slide Time: 14:06)



MBT advantages

- Model-based testing is easy to understand from both the business and developer communities
- Model-based testing separates (business-) logic from testing code
  EASIER SEGGREGATION AND INTEGRATION..
- Model-based testing increases the test coverage with the same effort as "Classic" test automation
  VISIBILITY IS LITTLE UNCLEAR - ESP COMPLEX SYSTEMS DIFFICULT TO HAVE COVERAGE CONTROL
- Model-based testing is the fastest way to get use of automated test

5

What are the other advantages model based testing has let us see model based testing is very to understand because the model is execrable that means use can see the picture representations and the as well as the complete representations of each of the sub model list that is comprise so basically it is very easy for tester developer or the from the prospective so basically the complex system we can offered to have some times spending.

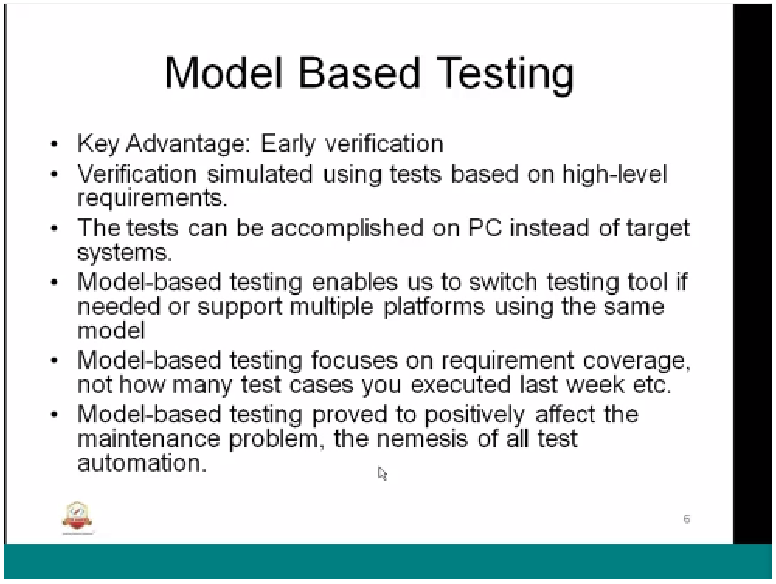
On developing the models in advance so they, they use this model based development system so this is useful because it is easy to understand from all the prospective designer as well as the tester primarily the developer and then the tester model based testing separates logic from testing code that means we do not to have code at the individual level we can have a logic at the business level that means the functionality has a whole and a functionality has a individually level.

And the code that we have which we want to test it that can be separated so it will be very easy segregations of the different levels so easier segregations and integrations so this are basic thing that will help in terms of separating it so there will be advantage model based inter the test coverage with the same effects as classic test automations so classic test automations we do not have model or model based system it is basically go by the requirement.

And the requirement patient that it will try to understand the functionality and try to fill the signals and all that so where the visibilities to mush unclear that means especially complex systems so it is very useful they will having a complex systems to have model based design testing in terms of classic automations the coverage is somewhat difficult it is called coverage control, that means we need to spent the lot of effort in terms of identify the coverage.

And to understand the coverage and reports but where as the model will basically help which model are done what are the signals covered so basically the concrete idea it gives for the coverage purpose model based testing is the fastest way to get use of automated test that means somebody to have a understanding and develop the automations it will be easier to have a model available for testing so that is the meaning of the fast advantages okay.

(Refer Slide Time: 17:35)



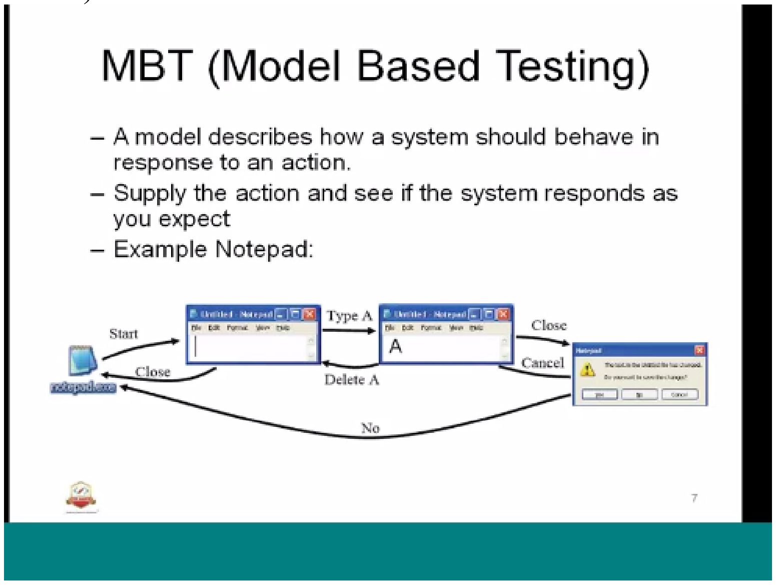
Model Based Testing

- Key Advantage: Early verification
- Verification simulated using tests based on high-level requirements.
- The tests can be accomplished on PC instead of target systems.
- Model-based testing enables us to switch testing tool if needed or support multiple platforms using the same model
- Model-based testing focuses on requirement coverage, not how many test cases you executed last week etc.
- Model-based testing proved to positively affect the maintenance problem, the nemesis of all test automation.

6

So the other advantage are something like thou the model not have been implemented completed there are design and other things that could been done already that without the need of the implementations somebody can start the test cases test scenarios their finishing that means the earlier verifications by seeing the models we can find the test loss that mean the test can find some of the mistake that the model has but visualization.
It and going through the model itself having his own sort of test cases identify so that is the advantages then verification simulated using tests based on high level requirement that means verifications can be simulated we do need to have a target actually we can do off line as I told in the next one the test can be accomplished on the PC instead of the target we do not to have a target based system in the beginning we can have the verifications done using simulated of the models.
For the higher level requirements we can basically have a coverage from the high level requirement from the models that we have under test model based testing enables us to switch testing tools if need or support multiple platform using the same models that means the model supports the plat form that is being deployed same thing can be used for testing also and the tools that are used it is very easy to switch that means the same model which are developed.
And deployed into the target with the help of the tools can be used for testing also so tha6t is the advantage of that okay so the next one is the model based testing focus on the requirement coverage not how many test cases you executed last week etc that means we do not want to care about testing and all that of course that is the different aspects but we o by model by model so the reach model we will cover as if the progress focus.
On the model you automatically, the more requirements automatically so that is the advantages last one being model based gas been improved to positively affect the maintainers problem nemesis of all test automations that means the maintainers of the models as well as the model based testing compare to the requirement based testing of without the model what we have what we can say as a classic automations testing.
Because the models one if changes with EF2 adopted to testing changes as well that is why model based testing is here to maintained compare to the classic automations type okay.

(Refer Slide Time: 21:01)



MBT (Model Based Testing)

- A model describes how a system should behave in response to an action.
- Supply the action and see if the system responds as you expect
- Example Notepad:

Start
Close
Type A
Delete A
Close
Cancel
No
notepad.exe
A

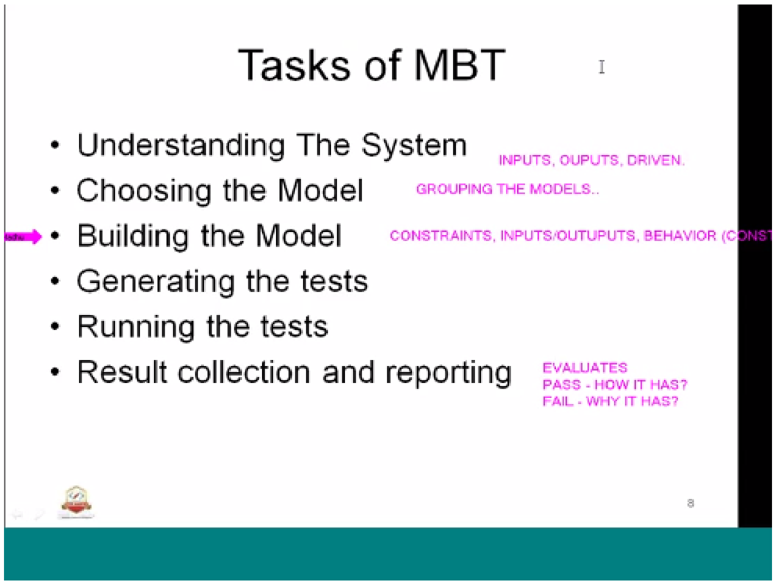
We will take a example responds from the one the good example from the web actually the model describes how should be there we can understand with the example below so we need to supply the actions and see if the system responds as you what you expect that means we have a state define as a blocks we can see there are four block below right so we take this as a examples where we have note pad, notepad is an example.

Here you can see we know all will know that how will you operate the note pad we have note pad executor opened we are going to start so there is a blank note pad that is going to open if you do not want you can close it so the operations that can happened the model start or close in next what we are going to do once we have started the model is that a able to type something and if you do not want to type that we going to delete.

It will be black to the blank sheet and once we have done we are going to close it ones we type it we are going to close it while closing it will ask whether to save it or not save it or cancel so cancel mean that will be back to the same state what, which is having it that means the third state or the actions will be done with the help of this transitions and we do not want to save it will be back to the note pad execution mode where the note pad ids closed.

So simplicity approach here we can see the model so the model should work something like note pad as an exactable should open should able to type should be able to close and cancel so we know that are the action that the system note pad system can responds the model is in try to sub set under test that is the fundament about model based test okay once we understood the about the model based examples with the note pad.

(Refer Slide time: 23:44)



Tasks of MBT

- Understanding The System — INPUTS, OUPUTS, DRIVEN.
- Choosing the Model — GROUPING THE MODELS..
- Building the Model — CONSTRAINTS, INPUTS/OUTUPUTS, BEHAVIOR (CONST
- Generating the tests
- Running the tests
- Result collection and reporting — EVALUATES PASS - HOW IT HAS? FAIL - WHY IT HAS?

8

We will try to understand what are the tasks of MBT model based testing so there about six steps or six tasks understanding the systems choosing the model building the model, generating the test running the test result collection and reporting the typical system testing what w have seen in class so similarly things we will apply here also where we do the model based testing first we need to understand the system here system and the models.

So we will try to understand the models once we understood the model we are going to start the test by choosing the right model that means which want to test which want to target first before the attack the sub models in terms of the top down approaches so all this sections criteria will be done with the help of choosing the model once we have this we do the built, built the model that means we develop the test model what we want to do it once.

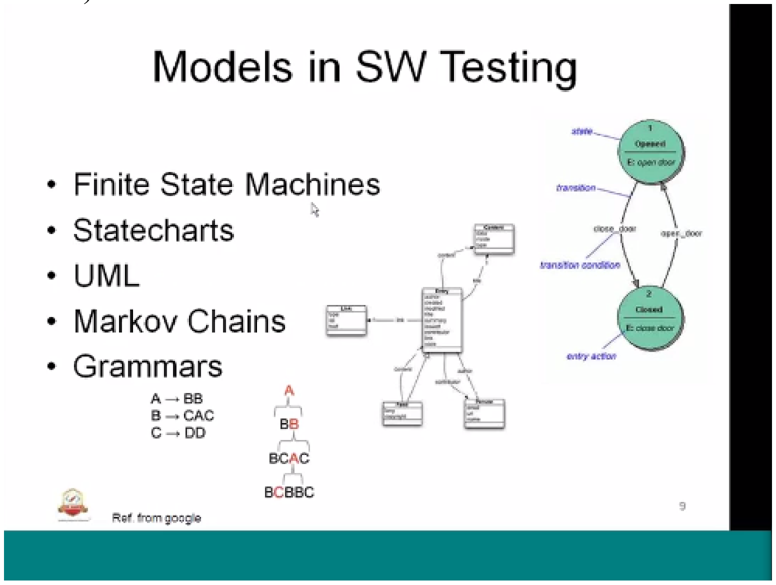
If we have that model that is ready for testing we generate the test that means all scripts for this model can be done or we will do a run through of this model with the help of various tools such as UML and all that then once we identify the test we are going to run it you see that how the model behaviors once we are executing the test owe have the result collection and reporting in terms of coverage and all that okay understanding the system basically forming.

Representations of the system functionality this is the small task because already we have the model and complexity will be explained in terms of understanding it so basically the software is developed for the particular model will be deployed on the target so all this information will be part of that model by understanding that model we now that what the model does so basically we determine the compacts' features what are need to be tested.

That is the objective of the particular test so while doing the model so we also have to established, commutations is called, commutations will be requirement with the requirement design development whatever it is that is very important we need to established the model mapped with the this aspects so it is very important along with that we are going to identify the impacts who all are there and we are going to interrupt.

So we will also identify the inputs, inputs outputs how it can be prevent all this aspects will be studied domain study it is called basically the number of document that we need to understand to map what we have understood the model basically so based on that we will right the test that is

why it is very important to have a understanding the systems next we choosing the model so basically what we do is we select which model how we can test.
It that means this basically help in terms of grouping because there are numbers of models we need to understand what are the models which can be grouped so that, so that can be tested or attack in terms of testing it so that is why the choosing the model is very important then we the producer for building the model this is also a very important step so where we will basically built that means we put the constants or inputs outputs.
All this will be laid out in doing the building of the model and of course behavior, behavior and behavior consistent also important so this are some of the building, building model aspects that needs to be taken care for doing the model based testing next we are going to have the generating the test so something like we have a machines or state diagrams so we need to have then, we have seen how worse path can be done.
And what are the sequences so there are different state charts state diagrams sequences diagrams activity diagrams so many are there maybe when we have a class for the UML we will study all of this that is of aspects of model based designing management those thing need to done in terms of the generate test because have understood that events and sequences we can have the test where to trigger and all then once.
We have that we are going to execute with the help of the script which can help in terms of executing those tests providing the inputs and running the conditions and providing the output also so those all some of the running details once we have the run details we need to evaluate whether the test are correct so it is very important to understand that how it has passed similarly to fail why it has so very important pass does not mean.
That it should be passed wrongly or it could have been get down the speed passed so how did we evaluate that so that is also very important sop we need to evaluate for pass and fail criteria while doing the model based testing so those are the task that we have for the normal testing.
(Refer Slide Time: 31:41)



Models in SW Testing

- Finite State Machines
- Statecharts
- UML
- Markov Chains
- Grammars

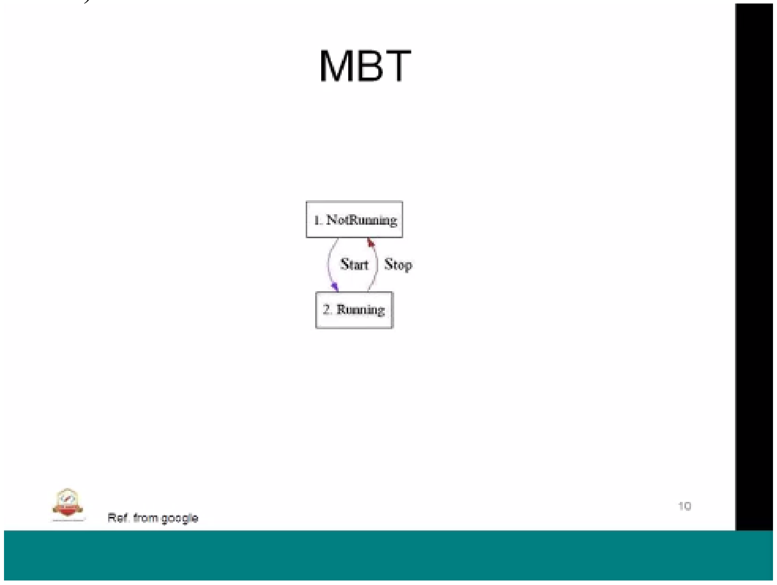
A → BB
B → CAC
C → DD

Ref. from google

9

We will take a one reference that I am showing on the slide examples of the model based testing so you can see there are finite state machines which has different state examples here and here

are given in the tool state two events specially one is door open and second one is door close the first is open the door and close the door so this is the action open door from close it will move to the open state similarly from the open state.
It will go to the close state with the help of close door action so this arrow is the transitions conditions similarly we use the state chart if you have see entry and there is contained so and what are the entry state in terms of various actions so there is author creator modify etc so there are lot of aspects that is involved in terms of which is called the task diagram this is done with the help of UML unified modeling language this the one of the very popular tool represented.
The models or use the models and we have other models, models based software also that is called markov chains, grammars which are bit complex in terms of understanding, testing we need to adopted any of the models going in detail and the, they will help in terms of model based testing so these are the examples of models that can be used it could be FSM state chart UML markov chains grammar any of those.
(Refer Slide Time: 33:57)

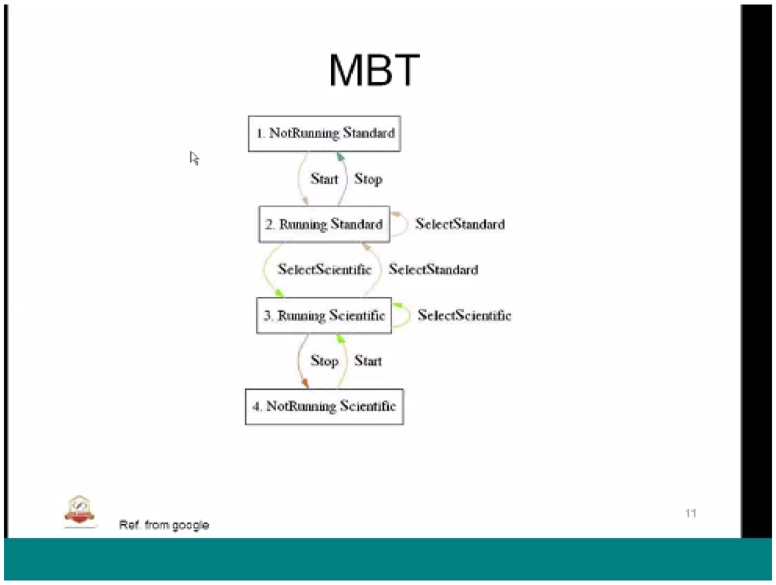


MBT

Ref. from google

10

So we will, we can take simple example management study we start with how we can starts with test the simple strict approaching that very simple is that you identify two state very simple first one is running and the next one is not running so the model can be entered with the running state the model can be execute with not running state so we will start not way that will help in terms of outstanding the complicity when we go in detail so not running and running the start will go first running stop will go into not running state.
So this two are represent the one model so how we can do a test he is with that he will apply conditions fuses good triggering not run to run so something like we will set alter or start as trigger that will enter into the running sp we will verify that whether the state has changed to running similarly the next test case will be whether it has run for whether it has change from running to not the not running with the help stop comment so two test cases we have identifies similarly we will see the next level of complicity.
(Refer slide Time: 35:21)



MBT

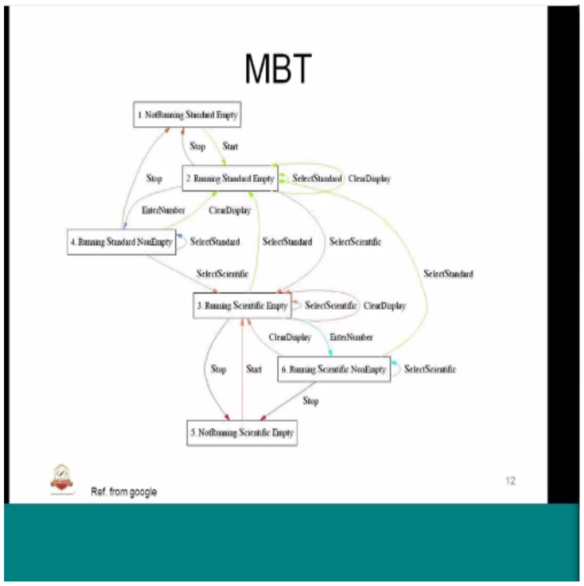
Ref. from google

11

The next you can see you can see there are about four state I thing we will give you an example of a calculator so what we can do is so what are the different state that calculator can have it take that as an example not running or a standard state is called we know that it will go to running with the help of the start and it will go back to not running with the help of the stop so two test cases can be there similarly for the second one running standard.

It can go further deep as a another running which is called scientific we know that calculator can operator will open it calculator you can see the calculator so we know that the calculator can run in standard as well as not standard way not standard could be scientific which we can choose with the help of the file edit and select the scientific so that is how we can do it here from scientific can go to not running scientific so likewise we can have a model based testing where we represent the model in terms of different state and how the model can trigger into different state we can understand with help of the standard example calculator going with the more complexity.

(Refer Slide Time: 37:52)

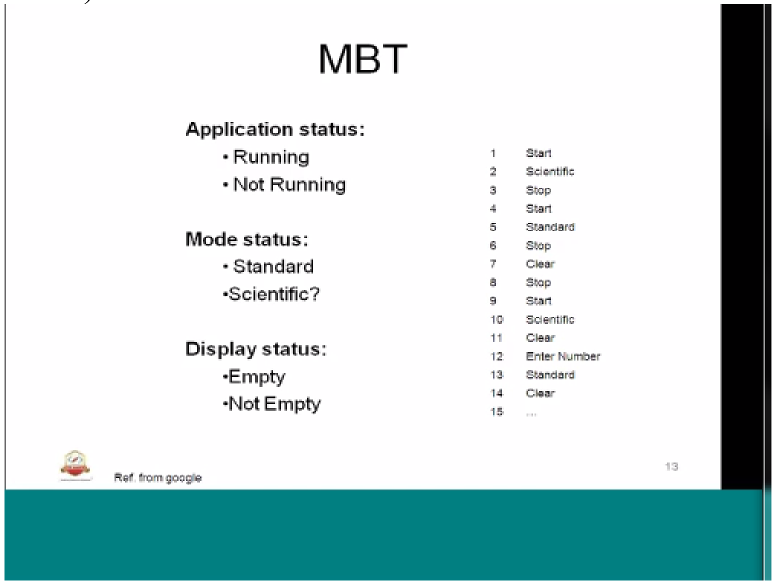


MBT

Ref. from google

12

You can see here not running, running and running standard empty running standard non empty scientific empty scientific non empty etc so each one has it is own sort of a conditions all this can be executed with the help of model based system where we apply the different state into considerations here you can see one set is there and other set of state conditions are existing likewise it can have the complex model based testing where we most text.
So here we have second one the next path it could be four or it could three and then five and go to six and from six it can be come back to one so likewise we can have you can see state separate it out. this is the original model this is the test model it is an exact coupling of the model so here we have the model here we have the MBT suppose that is one path where we have mixed so we apply the model base testing philosophy to apply all this test scenarios into the model where the model is recommend at the left hand side different box.
So that is the concept of model based testing so I will repeat it again all this model blocks can be tested with the help of different events where the events can be not running to running, running to scientific, scientific to scientific empty scientific non empty and etc so all thi9s five4 blocks in this complex example we can have and each can go into different path with the help of the model based testing approach simple it represented with the help of the model based testing model another model which is another the click of the original model we can see on the right hand side so that can be called as a model based testing come back to the original model okay.
(Refer Slide Time: 40:54)



MBT

Application status:
- Running
- Not Running

Mode status:
- Standard
- Scientific?

Display status:
- Empty
- Not Empty

1	Start
2	Scientific
3	Stop
4	Start
5	Standard
6	Stop
7	Clear
8	Stop
9	Start
10	Scientific
11	Clear
12	Enter Number
13	Standard
14	Clear
15	...

Ref. from google

13

So there are three aspects that we have tested applications status mode status and display status it can be many thing going by the example we can see the application is running or not running because there is primary states which was made with the help of the running state from the standard state in that way we know what is the status of the of the applications similarly we have a mode that also can be tested so what are the modes that can be triggered to enter into different status we can choose standard method or the scientific method.
So these are the primary of the inputs for testing the mode status you test cases are return in such a way then we have display status, display can have a A, B, C or in terms of extra decimal in terms of numerical that 0123..9 whatever remember and it also can show the +- the logical

operations and whatever it is so primarily the calculator example has three type of status against which that calculator mode will be tested so application mode status display status, running or not running more status and display status has empty and non empty for the right hand side.
You can see the various steps that are involved the start scientific stop start standard stop clear start spot scientific clear so this are all of the executions that can be done like a test scenario there are about 14 cases that has been listed on the right hand side all this can be applied you test one of this tree aspects or three features of the functions so calculator maybe having a requirement something like the calculator should calculator application should enter into a running status calculator should be able to not running status should be able to enter into.
So this is one requirement, requirement 1 requirement 2 so instead of writing calculator should go into running not so it is very easy to have model what we seen in the previous example with that help of that model we start with the simplicity approach then little more complex then the full complex approach considering the model array whole so that is how the model based testing is carried out okay on the right hand side you can see the different scenario have been listed out okay.
(Refer Slide Time: 44:16)

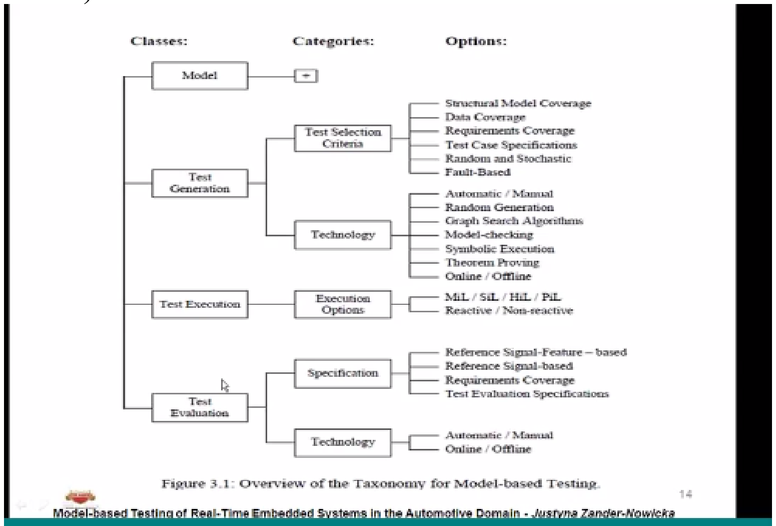


Figure 3.1: Overview of the Taxonomy for Model-based Testing.

The various what is that called taxonomy of model based testing this will cover basically all aspect for model based testing this is that what that is called complete model based testing overview this the gain described in detail in model based testing, in the automated domain, they is book called model testing for the automatic so that has a more discrimination about model based testing where he as basically have categorization in terms of class of and categories and option you can see the first column having the classes model test generations.
And test executions test evaluations so this are some of the class that are used for model based testing then we have categories model itself to the category where we draw the model next one is the test generations where we have a test selections criteria we know that test selections criteria can be based on the black box approach having the events or the to the various dynamic testing that we have seen in our earlier classes and that test selections criteria can be applied with the

help of this options column the test selection can be structural model coverage data coverage requirement coverage and fault based so all this are type of options.

That can be used for selections so with we are going to have test generations for this mode for this class test generations under class then we have a technology what sort of technology we are going to adopt for this test selections criteria as I said in earlier slides it could be automatic or manual a black box or white box when random generations graphs circles have got terms in model checking CRM proving, online or offline there are number of times number of options that can be used in terms of tools of the models next we have identify this test generations of the class.

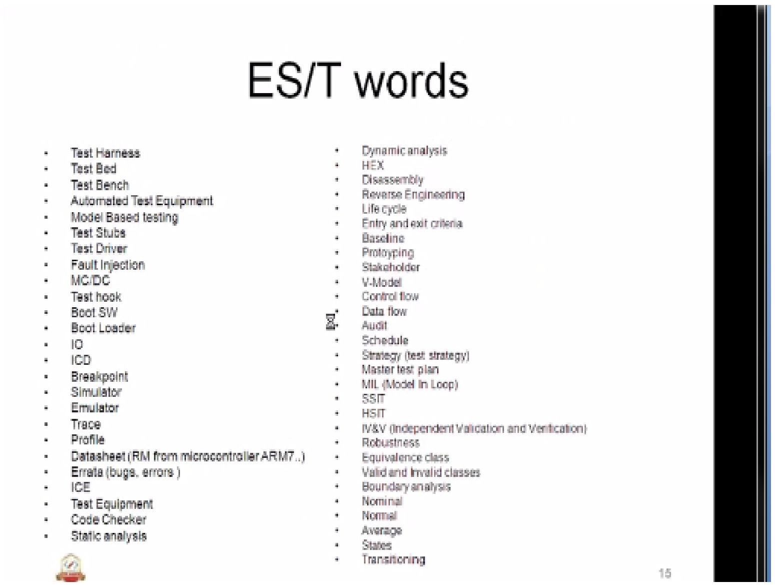
We go for executions class how we are able to execute so execution options are model in a loop software in the loop hardware in the loop etc so it could be reactive as well as non reactive now we are going to observe in terms of non reactive in terms of reactive we are going to incepts so that is one class then next class of the model based testing is test evaluations test evaluations has basically specifying the technology the specify will refer in mapping test that we have done with the above test executions of the we have the test generations in another class so what it helps is they refer all the features functionality that have been tested or test executed.

And all the signals whether that is covered boundary etc all this test selections criteria and execution of the will be specified and also that specification will have coverage in terms of how may requirement are have covered so what are the evaluations that have done for this coverage all this will be under the test evaluation class similarly the technology is always we are going to evaluate that means for example I am going qualified tools example I will take a automotive aerospace because we need to subjectively understand whether the tool is producing.

The rate result the intend result they should not have a any errors the tools should be error free all this will be part of this technology evaluations or it will values it could be automatic or manual it could be online or offline with the help of the vendor who ever has produced the technology or the tool output so that is about model based testing classes category and options which are used in the industry okay that is the end of this session there will be describe about models and task of the models, model based software testing examples taking calculator what are the different states.

It will enter and when we have the complicity we define the paths that models can take and we will replicate that using a model separately as a, so that is the philosophy of the model based system and we list out the feature vs. it is testing arrays categories in the model based exactions and we can have a class different for model based testing in term of the model the model, model category and test generations test executions test evaluations or the other class with the help of all this the model basis will be carried out coming to the embedded system models.

(Refer Slide Time: 50:04)



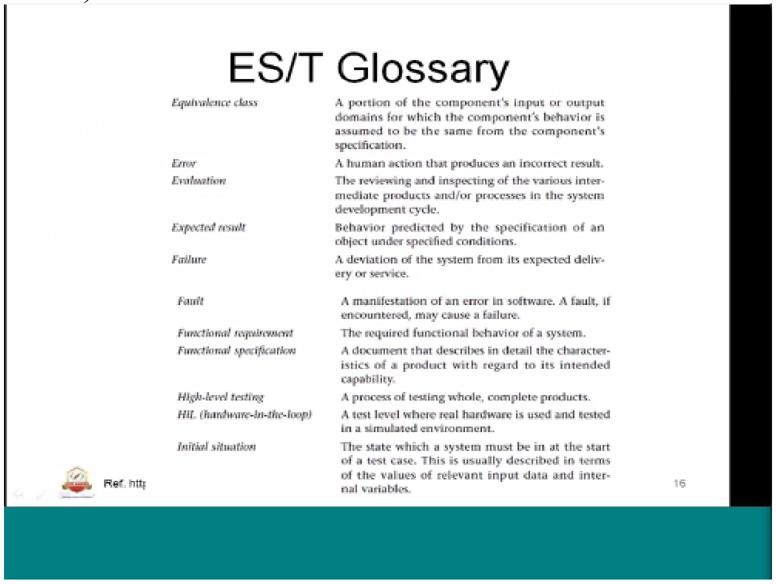
ES/T words

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

15

That we have gone through embedded system testing words so far in all the units test bench, automatic test equipments model based testing which we have studied today and next classes fault industry LCDC, test hooks, good software, good book, loader interface, control document, break point, simulators, emulators, profilers, or profile, data sheet, test equipment, code checker static analysis, static analysis, code checking in normal report of the next future session and in terms of embedded systems analysis we need to embedded system basic line remodel all we have studied will not go above in the one, software, software integrations test hardware integrations test normal domical average state transitions likewise okay.

(Refer Slide Time: 51:28)



ES/T Glossary

Equivalence class	A portion of the component's input or output domains for which the component's behavior is assumed to be the same from the component's specification.
Error	A human action that produces an incorrect result.
Evaluation	The reviewing and inspecting of the various intermediate products and/or processes in the system development cycle.
Expected result	Behavior predicted by the specification of an object under specified conditions.
Failure	A deviation of the system from its expected delivery or service.
Fault	A manifestation of an error in software. A fault, if encountered, may cause a failure.
Functional requirement	The required functional behavior of a system.
Functional specification	A document that describes in detail the characteristics of a product with regard to its intended capability.
High-level testing	A process of testing whole, complete products.
HIL (hardware-in-the-loop)	A test level where real hardware is used and tested in a simulated environment.
Initial situation	The state which a system must be in at the start of a test case. This is usually described in terms of the values of relevant input data and internal variables.

Ref. http

16

Some more glossary we study today equaling class we have studied in separate class input output for which the compounds behavior. Error a human action that could be an incorrect result a evaluations the reviewing and expecting of the various intermediately product and process of the system model test or development expected result we know that it is a behavior in predications of the specifications for a particular input under specific conditions failure a derivation.

From the system from expected behavior or service fault of an error of the software the fault is encounter then the result of an error basically functional requirement required functionality or the behavior of the system function specifications is a document basically that describe the characteristic of the product that regards to internal, high level testing you know as a complete product we are going to stick with the help of high level requirement or specifications hardware.
In loop where we put the test with the action target of the real hardware is used in terms testing and initial situations the system must be in the start of the test case that means the pre conditions are also called as it is very important is usually described in terms of the data internal variables okay that is the end of the this session of the embedded system so in the next session.