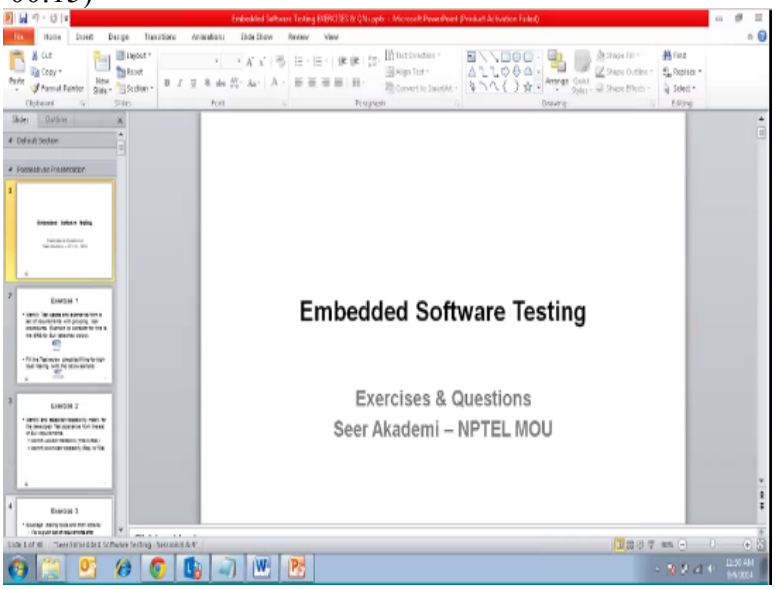
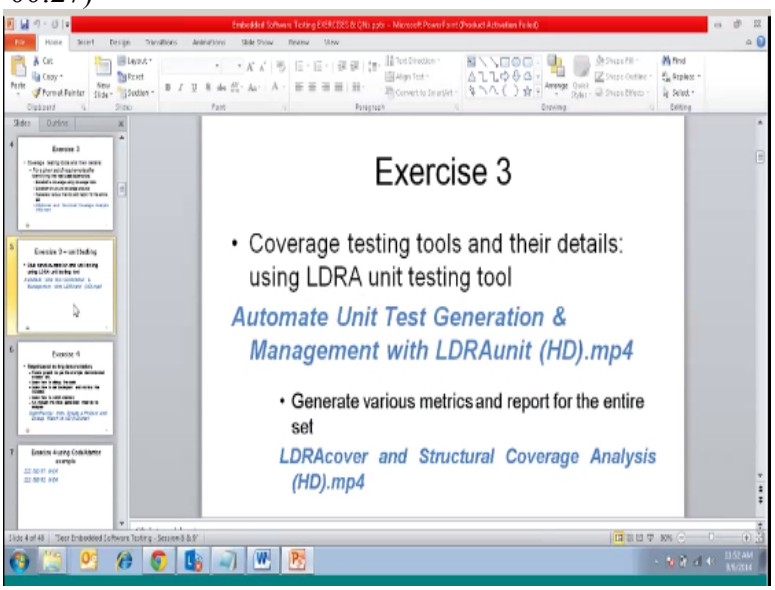
Welcome to the next practical session of vendors of testing today we will try to focus on.
(Refer Slide Time: 00:15)

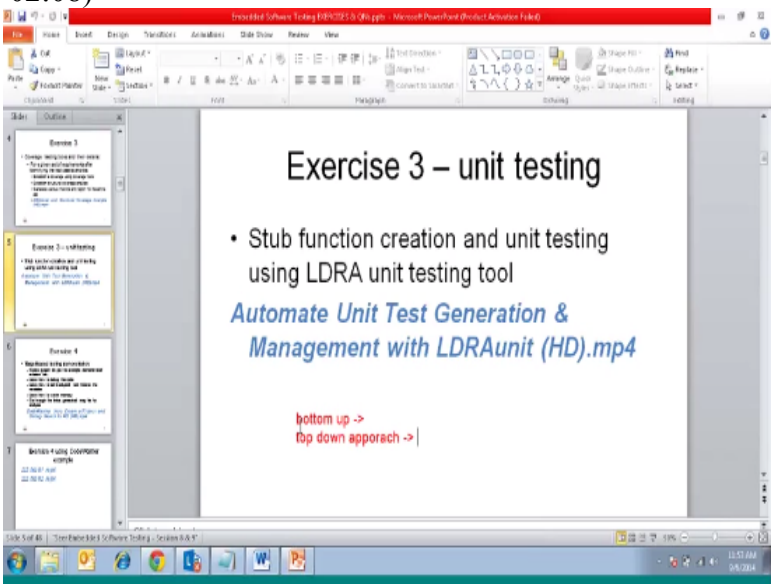


The exercise 3 that is on coverage testing tolls for unit testing so know that members of the testing
(Refer Slide Time: 00:27)



We have the life cycle such as remodel were right from system requirements to software requirement, is a requirement analysis high level design, parallel level design code on the once head and on the another side we know system valediction system integration hardware, software technician and software integration on the high level design and the unit testing.
In terms of detail design or low level design that we have and on the pair with this is detail design and the low-level testing we have the implementation of the code that is also required so that is what we try to do with the unit testing, so what basically we do with the coverage testing

tool is something like we start with the instrumented and non-instrumented method so the focus is to cover all the top level requirements.
That is implemented or no implemented then we will generate the report and see whether it is a covering 100% or not so that is what the purpose of the unit testing we have so basically objective of software testing demonstrated software satisfies the requirements so we will try to see how the tool is used example to let to be pickup today's idea relate it is a unit testing tool also followed by that we will tried to see the structural coverage report analysis how it gets generated.
(Refer Slide Time: 02:08)



So far I our char machine that were black box testing, white box testing in terms of testing methods we know that three levels of system integration testing and unit testing that can be done, so unit testing we will try to focus on first test units we know that we have stuffs and drivers that we need to live it, so we will try to touch base on that part. S
So there are bottom-up approach top- down approach , so top-down approach we are trying to test a higher level module with the help of stubs of the, the particular unit which is under the interest for us to hesitates right, so we have suppose F1, F2, F3, F4 four dot C files, or the C files data to be tested uniquely, and we are trying to test has a complete unit, so we will have to do, first one at the focus at the F1.C F2, F3, F4 C. Z stubbed because the function of interest for us, similarly that we are doing we are doing for F1.
So likewise we are going to have the stub creation, so likewise we have specific interfaces sometimes that also it to be addressed, so those interfaces are also stubbed something like stub system suppose so for unit testing the dependency on this sub system or sub units are interfacing

its, so we are going to have a stubs, and the stubs will have basically the parameters that are required to pass on and the parameters are the return values of the variables.
That are required to return, so with the help of that we will develop it, how it can be automated that also some space will take up if that cannot be automated then in that case we have to do a manual verification of the same.
So basically for low level testing what we do is the we looked for SDD or low level define also called software design document at its low level not the high level, because that will talk about all the lowest possible design.
Algorithms some Embay system will have pseudo code so in generic and that will be implemented in the code, so along with the SDD we also need the code.
Because instrument and non instrument excreta. So basically you try to cover all the patrician equation moldy reverences with all this ranges all that with the provided specification. And what we try to do he is non instrument target object code will take execute on the target basically reference other were the target based. Execution is attacking here that also we go through and all the requirement functional test will be done with the some limited values on the target board of the simulated target.
There actually this depended in a target at this period so it cannot are difficult instrument and robots also will do that and we take to get the result. Similarly on the course side we have the instrumented native is called has R instrumented course object code. Finally we had to compile the instrumented source code along with the original code to create target object code. And will be running that and will be passing on.
All the values that require for normal range Robots name excreta and the entire requirement and the compared the results along with the there one. So complaining these to gather complaining both this we going to come off with the coverage that what we do the coverage analysis with the both instrumented and non -instrumented.
In a testing okay so in terms of devotion that is the hero space standards it in standard that has to be adopted. So basically it is talked about spectral coverage and analysis and the objectives are will be ensure the tactual coverage that is each source of branch he is exsiccated at least once. That means we have source code with the branches and the condition right so every branch has to be in terms statement or decision has to be exsiccated. On the target or and the stub function whatever way want.
So along with the STD we also mean the code because instrument and non instrument excreta. So basically you try to cover all the patrician equation moldy reverences with all this ranges all that with the provided specification. And what we try to do he is non instrument target object

code will take execute on the target basically reference other were the target based. Execution is attacking here that also we go through and all the requirement functional test will be done with the some limited values on the target board of the simulated target. There actually this depended in a target at this period.

So it cannot are difficult instrument and robots also will do that and we take to get the result. Similarly on the course side we have the instrumented native is called has R instrumented course object code. Finally we had to compile the instrumented source code along with the original code to create target object code. And will be running that and will be passing on all the values that require for normal range.

Robots name excreta and the entire requirement and the compared the results along with the there one. So complaining these to gather complaining both this we going to come off with the coverage that what we do the coverage analysis with the both instrumented and non -instrumented.

In a testing okay so in terms of devotion that is the hero space standards it in standard that has to be adopted. So basically it is talked about spectral coverage and analysis and the objectives are will be ensure the tactual coverage that is each source of branch he is exsiccated at least once. That means we have source code with the branches and the condition right so every branch has to be in terms statement or decision has to be exsiccated.

On the target or and the stub function whatever they want so that execution try will be done with the tool basically will have test cases created for the requirements basically the requirement will tell about the values type.

Those values type this branches and divisions so the coverage and the robustness so we want requirement is that issue the statement decisions what are the conditions and the decision coverage that we need to take care to identify the dead code also one of the important objective.

So, statement coverage mission coverage modify mc DCD coverage that is independency of the different values at the input level, as well as the output level have to be at the end that what's we do the de modules the and also dead code.

This to be isolated it should be removed basically and it must be performed to as the affect and the need of 2nd time verification so we have to remove the dead code and are we really requiring to execute.

Remain require do it because regression test necessary that case were the impacted a at as to be re verified or re tested, so some of the example of robustness are something like a /0 it is the one of the were the how the code it to be the behave for this situation and to verify the component behave require of the over flow for a local data all is going to behave, something like a negatives

for rich fair route are some impossible operation will as per try at any case software. Should away offer a part known are determination.
And determent execution that is what we need to verify so that things to not with a specify in the design document or the lower design document they should be a figure out, okay so basically the lower testing and lower level entity most independent unit this can be modernized able to add it we need to input at detailed design at the detailed design and the testing process is unit testing it will be a test report and the coverage report.
Which will tell how much covered in structure branches statements so basically we trying to cover the functionality of the model at the same thing it is robustness also will be verified? So to first way to do this is using the cod itself you do it, so it is not just a try to do the unit testing help of this code with the only defined.
So the main input for a test arrived to perform the unit test in these software design document and not its hotspot basically is a document in the input for in based on that he has to try the value. Where he is going to try the value? He is by input has code so with the code has input the requirements are not tested and dispatch between detail desire code cannot be detected sometimes. So unit testing we may have to take care for those situations okay.
So there are different tools like RDRT, fantasia, letter cast, CTB, LDRA. So we will try to go through the LDRA tools how their uses, because this all available in net actually. I have downloaded and we can use as alignments
(Refer Slide Time: 16:20)