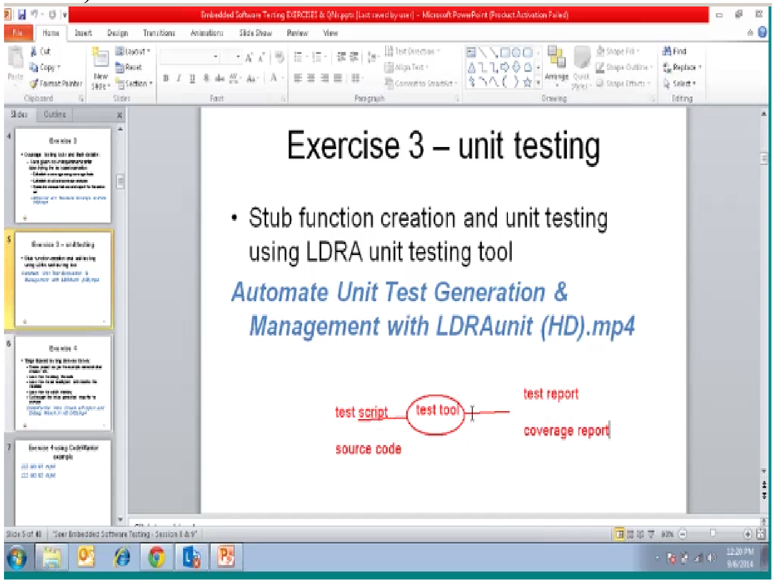
Okay so in order to have the unit testing used.
(Refer Slide Time: 00:13)



Exercise 3 – unit testing

- Stub function creation and unit testing using LDRA unit testing tool

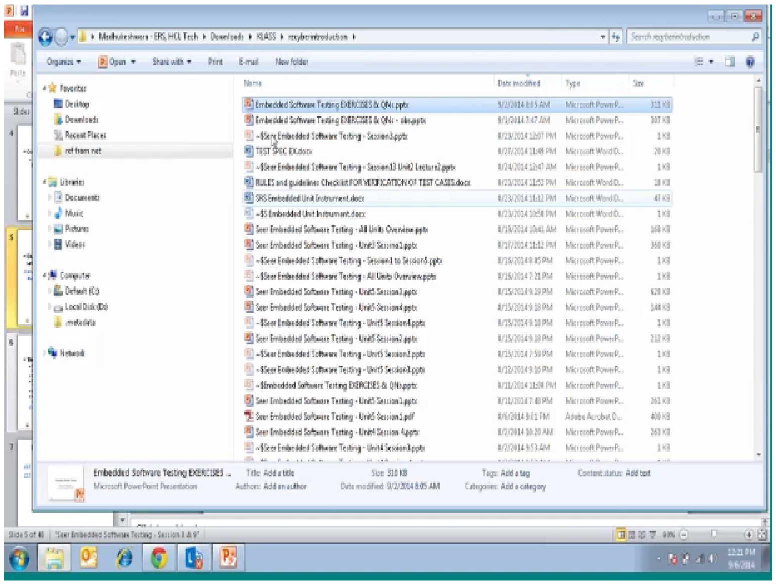
Automate Unit Test Generation & Management with LDRAunit (HD).mp4

Into have reddest tool basically so what are inputs that we need, so one is that we need test scripts other one is the implementation of the source code, so what is the output that we except on the test tool the unit test tool, test report till generate the report and automation if we have with tool we use coverage the report also to give how much % and all that basically this is an optional we can do it manual also.

But this tool requires the test scripts and the source code the test is something like the inputs that are required and value it drives and test tool that and automates and generate the reports that is hoe it is getting used the test scripts basically derived of the plan, so it consist of tester inputs as the parameters and expected results in the script, the script is a possessed by the tool on the converter into a driver.

Basically driving the source basically so in tool like RTRD the name it is called as a PTO the driver name similarly earlier it is on format test report is the result of the test scrimption either it is coverage report how much of total source code have the covered by all, the test cases that is what it aspect okay. We will try to understand the tool what is does and go through some videos.

(Refer Slide Time: 02:29)



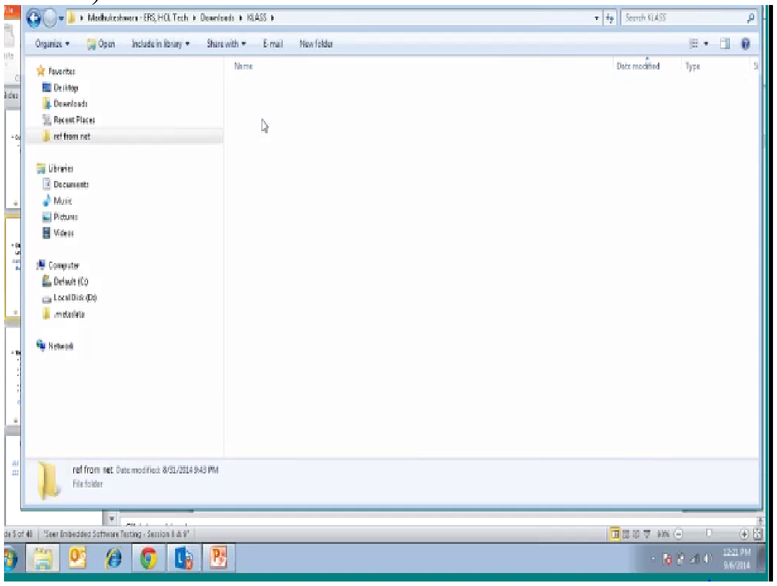
Madhuleshwara - ERS, HCL Tech ▸ Downloads ▸ KLASS ▸ rexybenintroduction ▸

Organize ▾ Open ▾ Share with ▾ Print E-mail New folder

Name	Date modified	Type	Size
Embedded Software Testing EXERCISES & QNs.pptx	9/2/2014 8:05 AM	Microsoft PowerP...	311 KB
Embedded Software Testing EXERCISES & QNs - obs.pptx	9/1/2014 7:47 AM	Microsoft PowerP...	307 KB
~$Seer Embedded Software Testing - Session3.pptx	8/28/2014 12:07 PM	Microsoft PowerP...	1 KB
TEST SPEC EX.docx	8/27/2014 11:49 PM	Microsoft Word D...	28 KB
~$Seer Embedded Software Testing - Session13 Unit2 Lecture2.pptx	8/24/2014 12:47 AM	Microsoft PowerP...	1 KB
RULES and guidelines Checklist FOR VERIFICATION OF TEST CASES.docx	8/23/2014 11:52 PM	Microsoft Word D...	18 KB
SRS Embedded Unit Instrument.docx	8/23/2014 11:12 PM	Microsoft Word D...	47 KB
~$S Embedded Unit Instrument.docx	8/23/2014 10:58 PM	Microsoft Word D...	1 KB
Seer Embedded Software Testing - All Units Overview.pptx	8/18/2014 10:41 AM	Microsoft PowerP...	168 KB
Seer Embedded Software Testing - Unit3 Session1.pptx	8/17/2014 11:12 PM	Microsoft PowerP...	368 KB
~$Seer Embedded Software Testing - Session1 to Session5.pptx	8/16/2014 8:05 PM	Microsoft PowerP...	1 KB
~$Seer Embedded Software Testing - All Units Overview.pptx	8/16/2014 7:21 PM	Microsoft PowerP...	1 KB
Seer Embedded Software Testing - Unit5 Session3.pptx	8/15/2014 9:19 PM	Microsoft PowerP...	628 KB
Seer Embedded Software Testing - Unit5 Session4.pptx	8/15/2014 9:18 PM	Microsoft PowerP...	144 KB
~$Seer Embedded Software Testing - Unit5 Session4.pptx	8/15/2014 9:18 PM	Microsoft PowerP...	1 KB
Seer Embedded Software Testing - Unit5 Session2.pptx	8/15/2014 9:18 PM	Microsoft PowerP...	212 KB
~$Seer Embedded Software Testing - Unit5 Session2.pptx	8/15/2014 7:59 PM	Microsoft PowerP...	1 KB
~$Seer Embedded Software Testing - Unit5 Session3.pptx	8/12/2014 9:16 PM	Microsoft PowerP...	1 KB
~$Embedded Software Testing EXERCISES & QNs.pptx	8/11/2014 11:08 PM	Microsoft PowerP...	1 KB
Seer Embedded Software Testing - Unit5 Session1.pptx	8/11/2014 7:40 PM	Microsoft PowerP...	261 KB
Seer Embedded Software Testing - Unit5 Session1.pdf	8/6/2014 9:01 PM	Adobe Acrobat D...	408 KB
Seer Embedded Software Testing - Unit4 Session 4.pptx	8/2/2014 10:20 AM	Microsoft PowerP...	263 KB
~$Seer Embedded Software Testing - Unit4 Session3.pptx	8/2/2014 9:53 AM	Microsoft PowerP...	1 KB

Embedded Software Testing EXERCISES ... Title: Add a title Size: 310 KB Tags: Add a tag Content status: Add text
Microsoft PowerPoint Presentation Authors: Add an author Date modified: 9/2/2014 8:05 AM Categories: Add a category

(Refer Slide Time: 02:30)



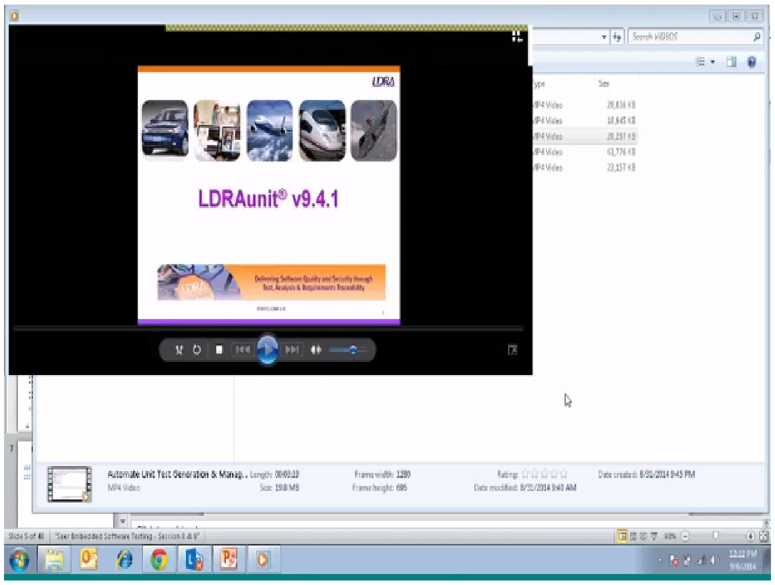
Madhuleshwara - ERS, HCL Tech ▸ Downloads ▸ KLASS ▸

Organize ▾ Open Include in library ▾ Share with ▾ E-mail New folder

ref from net Date modified: 8/31/2014 9:43 PM
File folder

In terms of demonstration okay, first we will go through the other unit testing generation management.

(Refer Slide Time: 02:51)


LDRAunit® v9.4.1
Delivering Software Quality and Security through Test, Analysis & Requirements Traceability
Automate Unit Test Generation & Manag...

Okay so we will understand the tool called LDRA unit basically this is the tool form LDRA and the version 9.4 .1 so basically we uses for unit testing and the source code we will tried to understand what it does this is a window where we have unit explorer so we will, try to explore software testing aspects of this regarding so there is a feed code selected with this end array here we have got a file called cash registered .C so we want to do some unit testing for this cash registered .C C 5.

So you can see the file contents it is explores basically ,you can see there are number of functions like add product ,account products emission etc, and sit here. Okay any function we want estimate alone like add products, we can see the parameters it has the variables are picking age and whatever the function it calls it always here ,LDRA unit 2 it explores basically what is individual function of this cash registered file you can see the structure passed as a parameter.

And it is calling Tested Show, Tester show and Sprinter three function it is calling basically, okay let us to see how sequence is created this two, need to select the sequence and we can call it as a unique tester add product something like this, you can add other sequence and we are trying to do a code coverage for a particular function with the sequence, we have to select the code coverage option then test environment something like setup.

About stuffs should create right, because it has number of function, if you want to call directly the functions it is available you can use it, if you do not want you can create the stuffs for those callers, so we can select that automatically create the stuffs with that option and user variables can be global if it is missing we can create it generate it and different type stuff what a scenarios we can hide here.

First option is allowing the compiler code as number one or the next one is creators for the subset of code for the search file is specific usage, because code the stub ways just on the lay we can do this or isolate fully all, we can do this second option just to create that particular single sort file, so let this function is selected, okay once you select the procedures it will show that particular procedure and you can set procedure name.

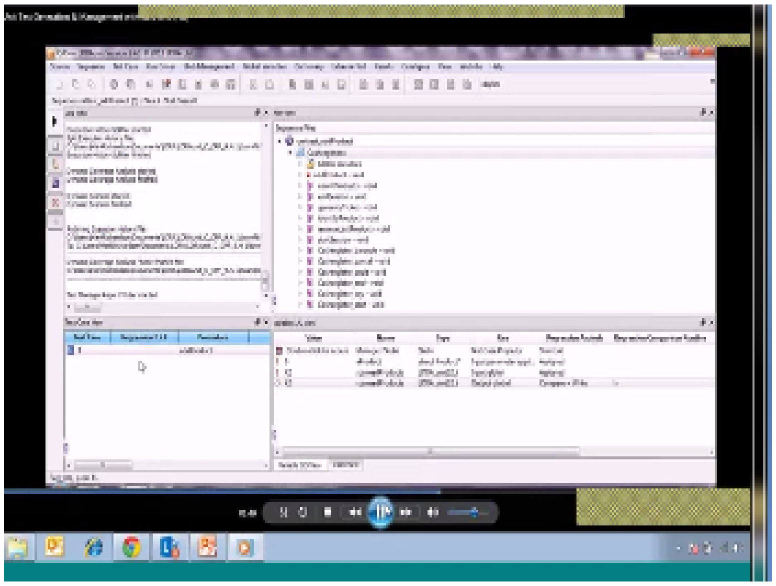
And the function and the other details we can stay okay, and we can continue, so it does the basic another of this option, here so what is going to happen is you can see the option here, the variable view and the calls will this option, and any stuffs that is has created it will show here it is tested, tested show number of calls is two it is a manage stuff, so here we go to a function and we will right click and create a test case that particular function.

Now and it is doing a instrumentation with this and you can see inside about to create a test case base up on the interface whatever displaying below, so here a test case generation is it could be standard single test case like multiple tension has to execute with different things, last one is important here recreations, sometimes what will happen is want to see the constancy with something like a sorry regulation.

So in that case we have a reputation. Are some ranges we can have it related by having, so you just try to go with the single test case, the music continue that here you can see the input output view, so there are two inputs for this function, the structure is a value I am saying in thirty two bate, structure and output expect as a much, so each an click base value input and enter rate enter at zero and the next one we enter it has forty two and the output we can expect put has forty two.

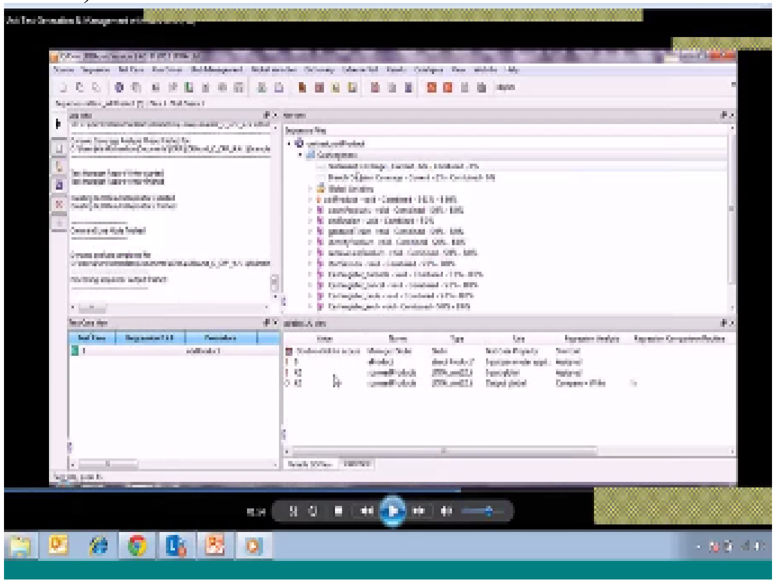
These are example now greater than this one you can see the generated test case and that is value, you can see here value for storage at the function it going to use this and the past way result bases the expected value of sources actual value. So the organizes generated with this that exsiccated and showing the result. So say okay then we can generate the again if you want with the multiple impose like 42 font change with to 50.these are the function are the inputs for the function.

(Refer Slide Time: 10:42)



Of course the input functions these base on the specification on the requirement so that will get proper coverage there multiple statement.

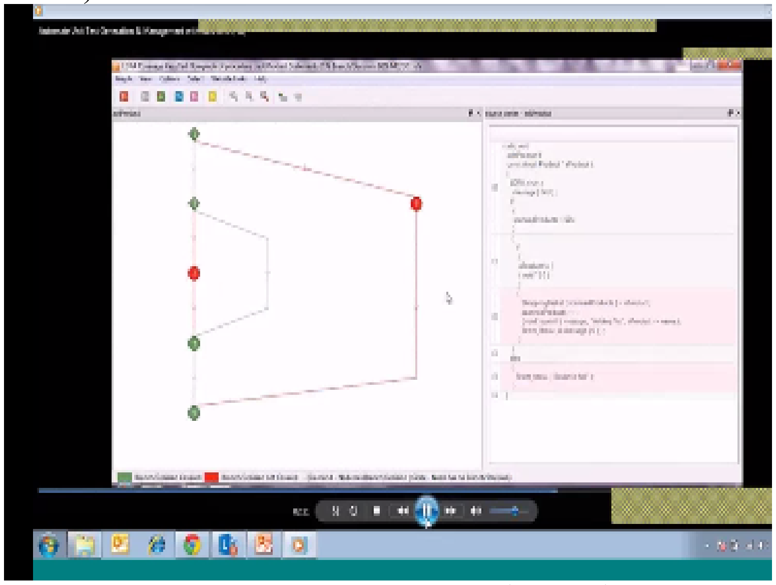
(Refer Slide Time: 10:51)



Within the function has been regard and we can see here that the coverage statement coverage the branch decision coverage. So current one is 6% compiled is commending actually this discussion. Current is 6% for grant of decision for that particular file entire file. And that function you can see statement for a 61% data speed at the branch has 60% we can try it the second time, second time again.

It should say has it been layer so it is a recreation. You can see that the recreation hash of fail it shows and this is change in the percentage there that the values are same. It has passed then the coverage related to look in particular with the flow graph of the coverage of the particular function. These interesting will see we can look at the flow graph see dish here this flow graph basically saying that exsiccated two times this block exsiccated ends all statement are covered in the block.

(Refer Slide Time: 12:25)

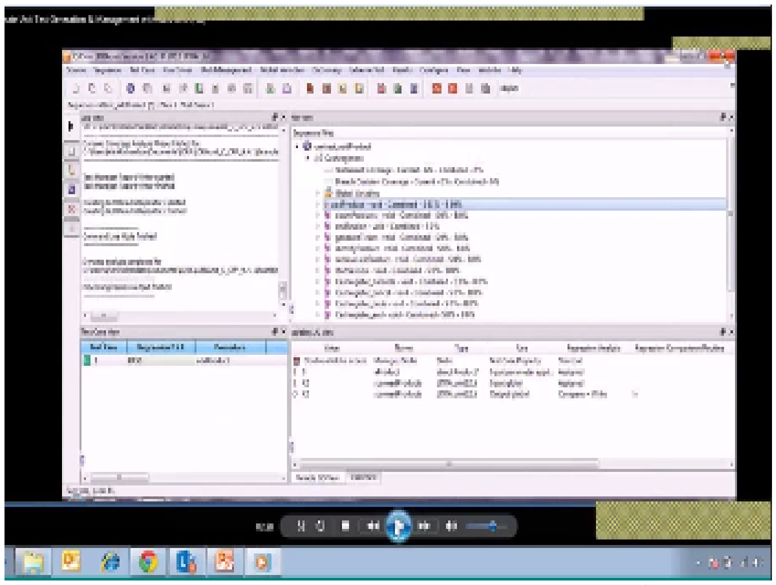


There are 6 blocks and that is taking this part after this 4th and 6th so basically there are 4 blocks that it could be exsiccate properly and two blocks 3 and 5 are exsiccated that is showing to red. On the red and side you can see that red mark one which is not executed with the part of the L portion and it is not executed. So does passed this one the portion that is not executed red one is statement is not executed.

Because the value that we are given is not in half to executes. 5 blocks is the last one rest for show last it 4 something like this so this useful to see the flow graph to the coverage. So and generate the test causes more causes so we can see it is adding 2,3,4.5 other 5 both the causes automatically and it will adopt all that test causes and generate the now with that let us see what it is generated.

You can see these additional 4 causes automatically it has assumed dated. So % with the 4 causes all to gather use that 100% for that particular formation and for the particular 3, 4 you can see the value.

(Refer Slide Time: 14:07)

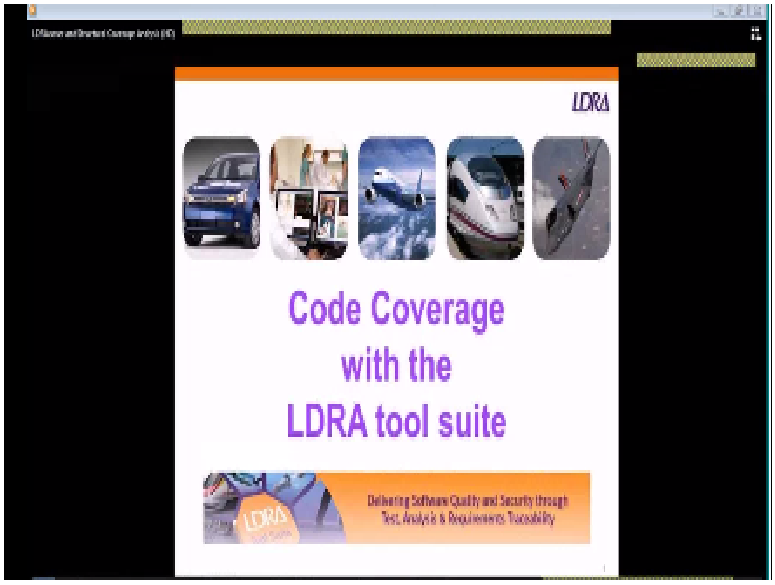


It has used based on the inputs that it takes so you can see the 4th cases and find the address passed for the structure 1st input the 2nd input is a global variable. And the value of 50 and the result you can see is that statement coverage's 100% and 100% branch it is covered. So this 4 we of this we can see that is the LRA unit thing has still we are not covered 100% of the entire file. See for because we are focused to under particular function of the file.

So that is the role earlier as b used, further information we can look into idra.com and a there is a evaluation caution probably you can download at the experiment more and that software testing and the aspects, so that about the tell here, how automate the unit testing generation is done of which you can see a %, and automatic unit test generation it does next type of IDRA unit is called a coverage.

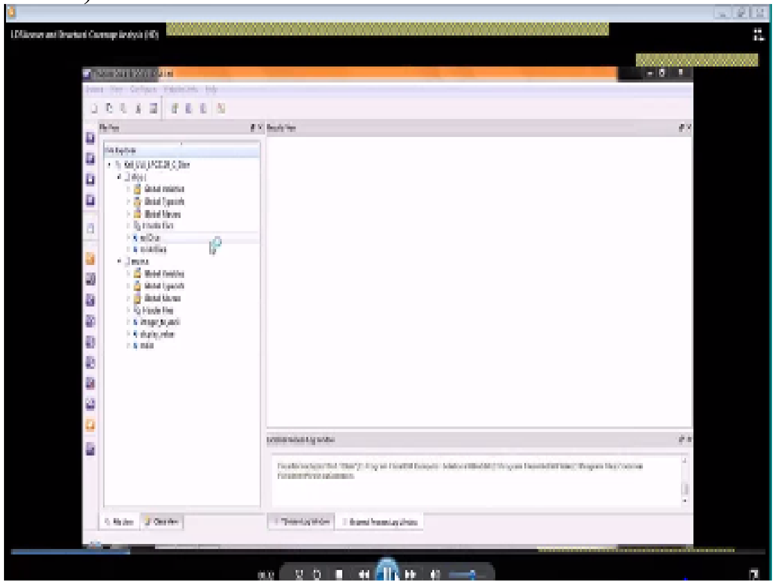
Also you can complete coverage of this source port and these structural code analysis and for a envoy software which as a automotive and aerospace let us try to understand that also, let us try to play a demo video for that we are required and understand, This called a tray aspects to suit to integrated together, so could coverage iterative understand.

(Refer Slide Time: 16:29)


LDRA
Code Coverage
with the
LDRA tool suite
Delivering Software Quality and Security through
Test, Analysis & Requirements Traceability

So here you can see an example of a project for a key trace at to r based target code we will try to similar so we do not have target code is actually connected this to and without we trying to have the coverage executed.

(Refer Slide Time: 17:00)

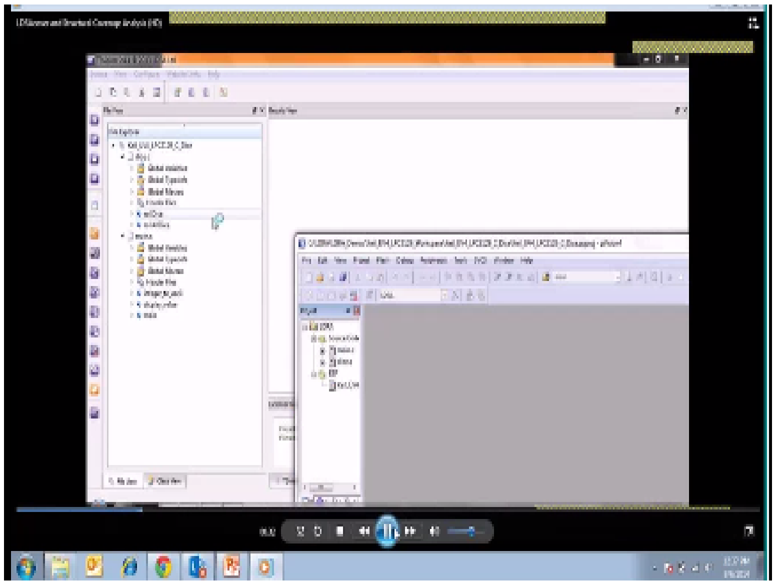
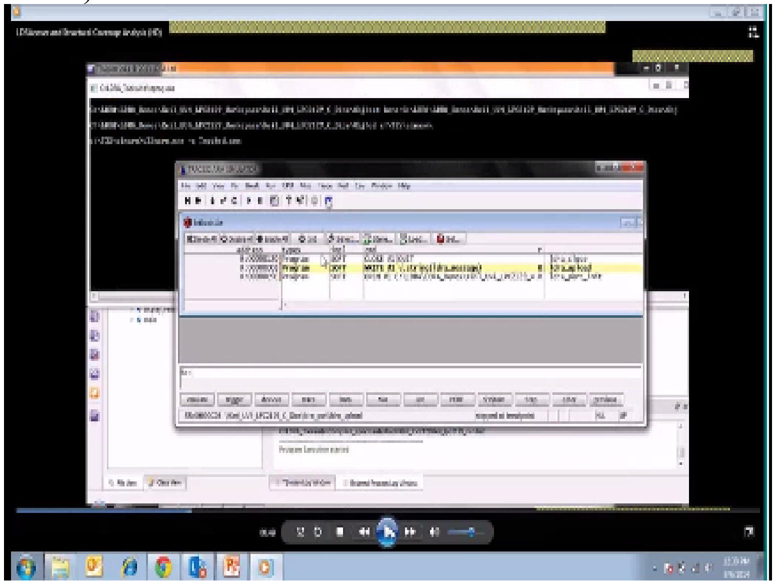


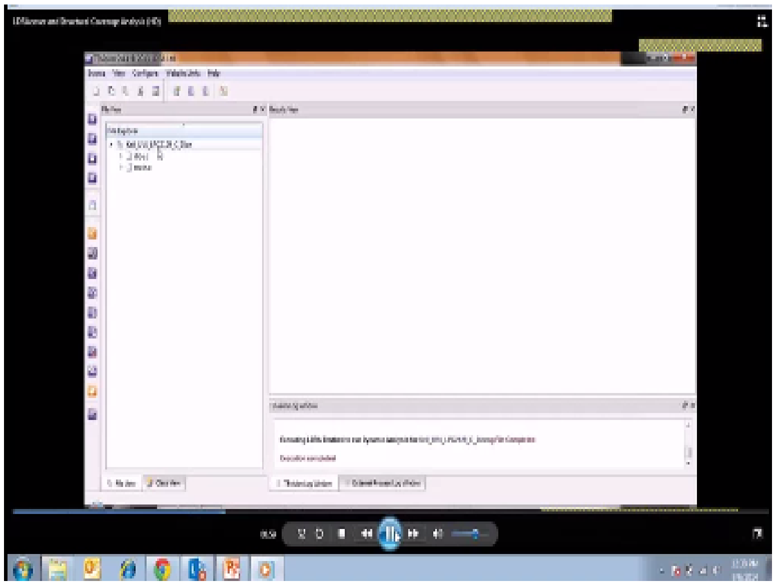
So it is a CPU based c file so this called as a TB vision in the array, so files spread on in the project explained here main. C price C simple files of the, so we will execute the code on the target here we do not have the target we can use the same letter we used it based on the target because due to that placement of the target so this is button we have to press that it is going to compile this.

And add the instrument the source code for the target code and this is going to build.

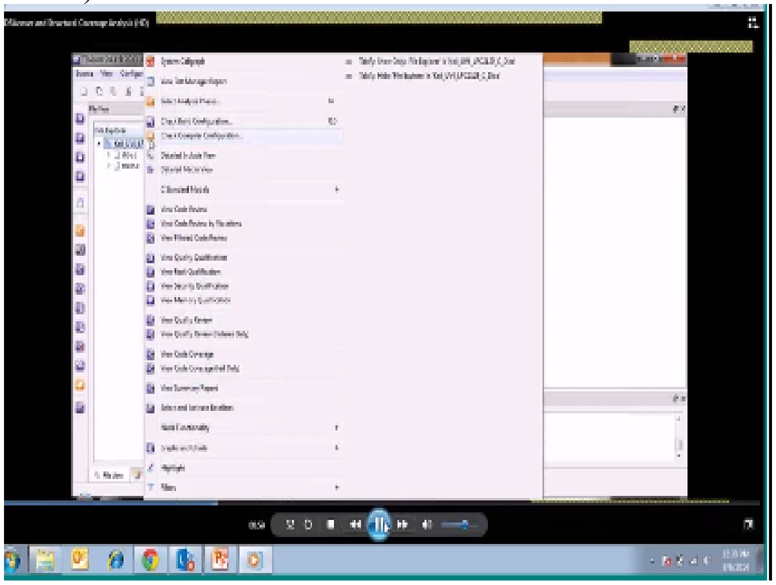
So it is a compiler it has been built is called a fail compiler in c of the elder test but is integrated.

(Refer Slide Time: 17:55)

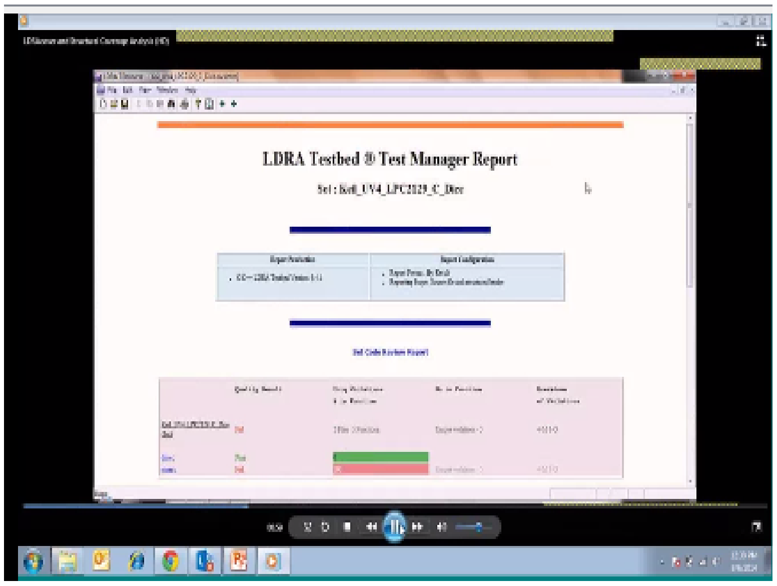

We can see it is running for that particular to the simulator right.
(Refer Slide Time: 18:05)

So what is trying to do is as soon as the simulator is trying to say a message on to the simulator continuously capture the unloaded value you can stop at some stage we can hard it is option and we can upload the result which is on that, the uploaded execution history can be see here we can see for using process for that.
(Refer Slide Time: 18:48)


And we can have a look at what coverage it has generated we can see the test manager report for this, so you see their report for the same.

(Refer Slide Time: 19:10)


We can see the code review different types of review it has what we are done here.

(Refer Slide Time: 19:17)

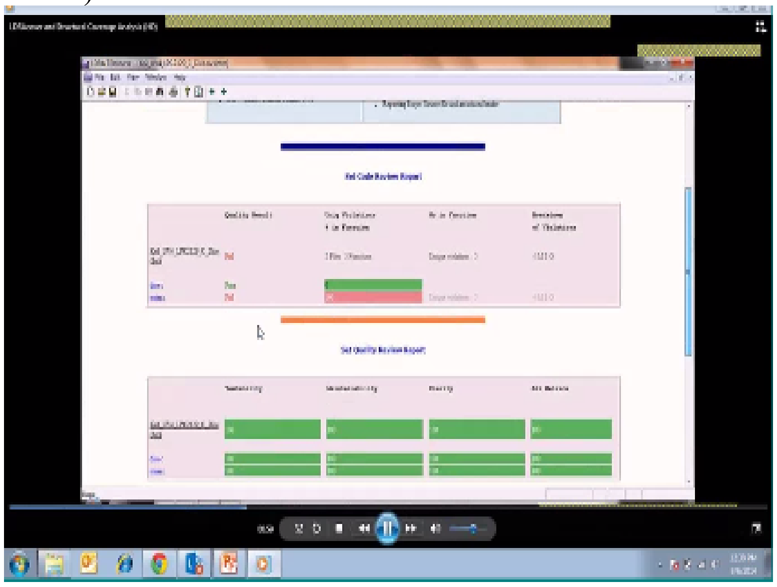

LDRA Testbed ® Test Manager Report

Set : Keil_UV4_LPC2129_C_Dice

Set Code Review Report

It is a testing location the dynamic report and we can see.

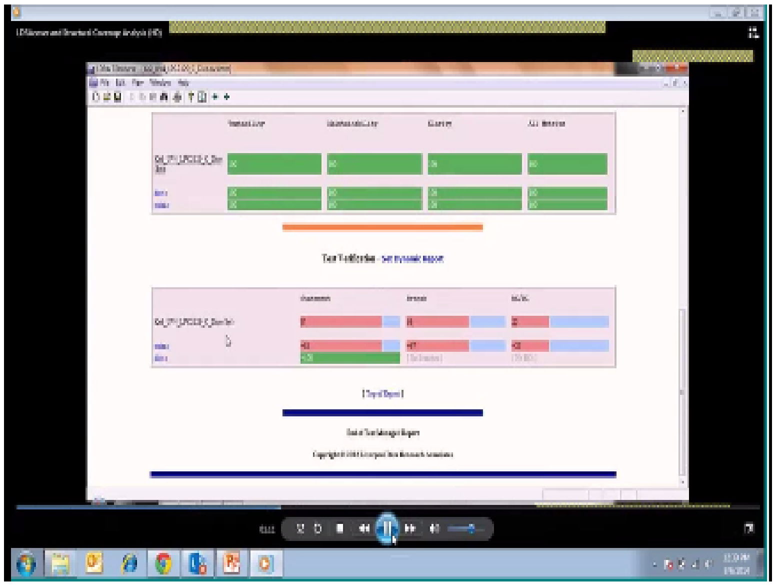
(Refer Slide Time: 19:23)


Set Code Review Report

Set Quality Review Report

And a file we have to try executing main. C and file .C

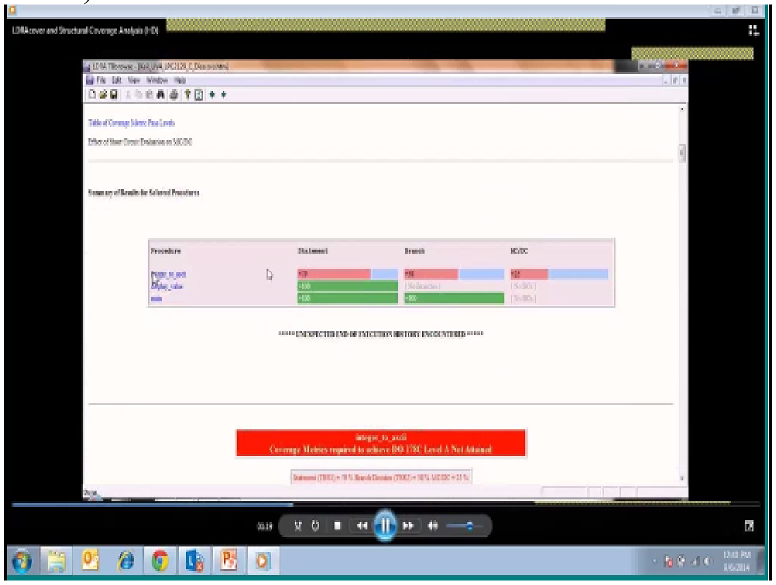
(Refer Slide Time: 19:29)



And it can be coverage of statements branches execution the choose an option see the % were it is 100% showing in the screen whereas are shown as r write so this is coverage is not 100%, let us try to go insert the particular file such as main. C we can see the main.c, report in particular, showing the result in red because, the coverage is not attained, whatever they obtained value, is not having 100% coverage because, it is 84% and blank decision is 67%, and hdc it is 75%, that is why it is not used, in the complaint, why it is not compliant, let us try to understand, we can scroll down and see.

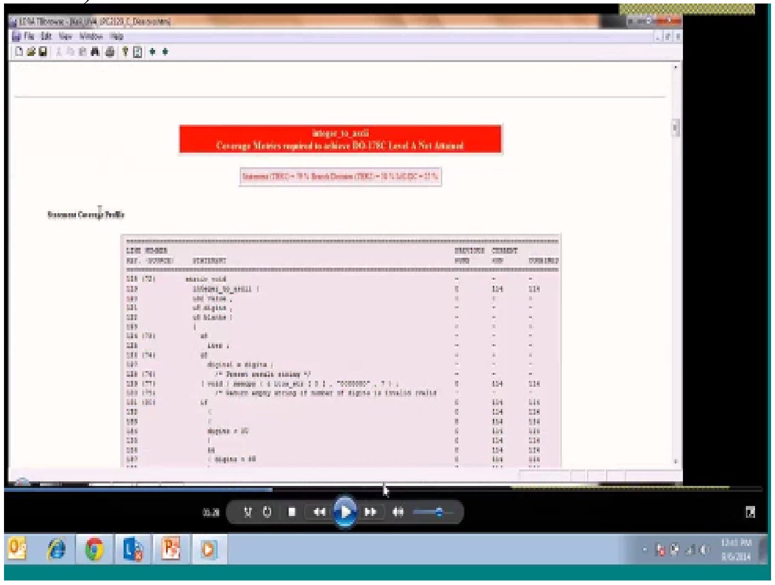
The main.c has total functions, such as procedures, such as integer, display value, etc and main itself two type of tests like.

(Refer Slide Time: 20:41)



Segments and transfer, which uses and 100% coverage is there, whereas the integer in built Transkei library, or the function, has at least 99%, and the coverage that we obtained is for this particular integer Transkei is statement is 79%, branch is 58%, and you can see that particular function, as the integer and if you see the last 2 theorems, current run, combined run how times it has run.

(Refer Slide Time: 21:27)

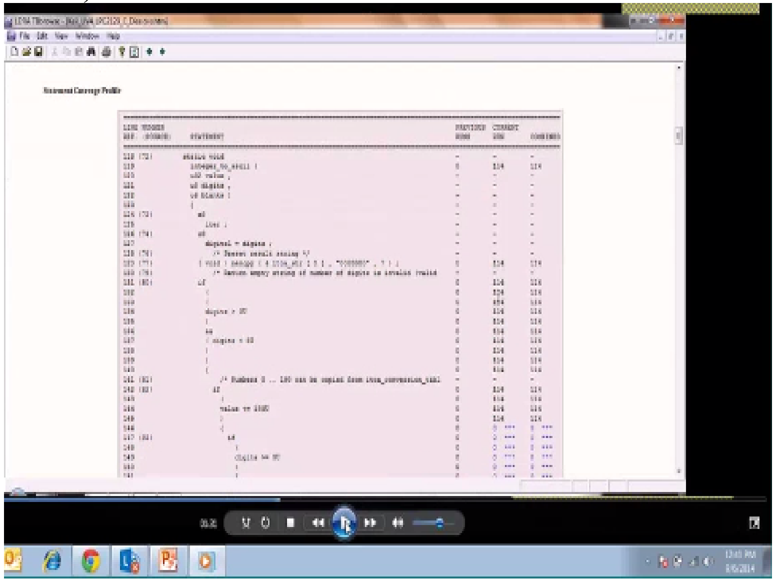


integer_to_ascii
Coverage Metrics required to achieve DO-178C Level A Not Attained

Statement Coverage Profile

Because simulator and it will show the number of calls, what are the statements that are being called, and how much decision it has, been able to drive together, put the every line of the code it is showing.

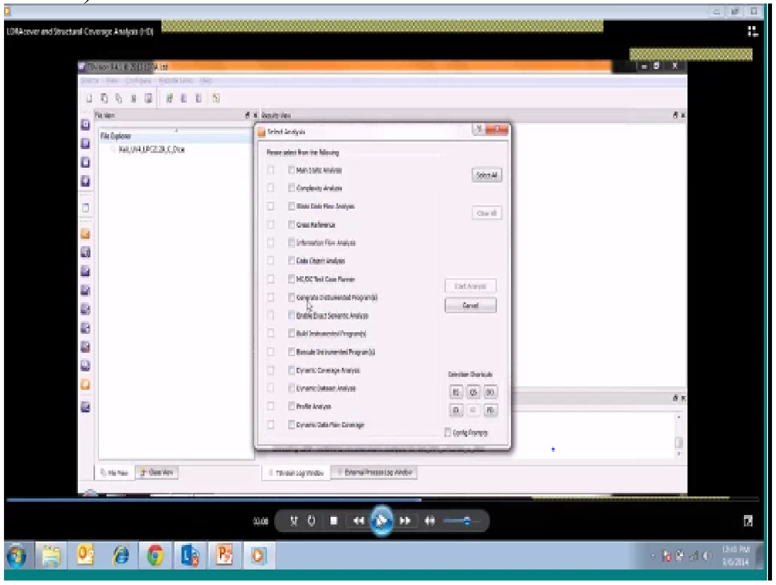
(Refer Slide Time: 21:45)



Statement Coverage Profile

So you can see the stars it is showing, because the values, that it requires, has not attained anything, so some of the lines ,we have executed more than, the earlier ones, like 500+ times it

has been executed, some of them have been executed 114 times, likewise, okay that is how we can see the report.
So one more approach is also, there we can delete all the results we can change the intimation technique, with other option.
(Refer Slide Time: 22:47)

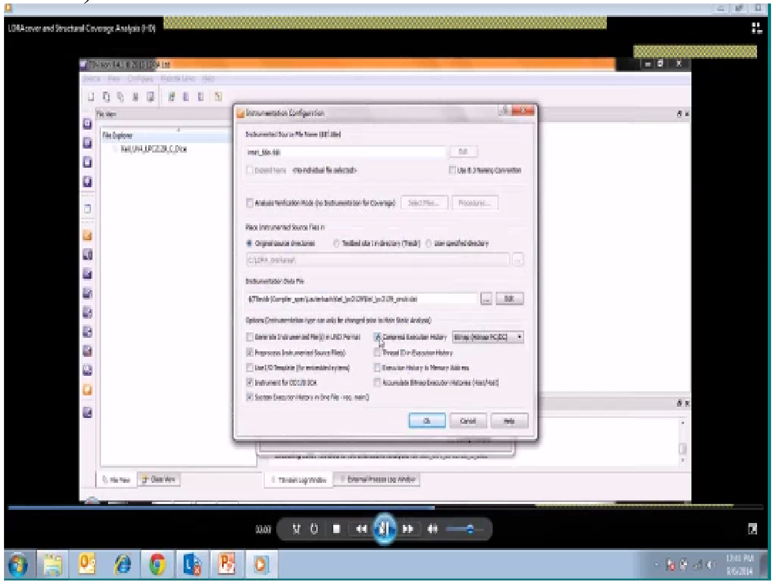


Select Analysis

Please select from the following

- Main Static Analysis
- Complexity Analysis
- Static Data Flow Analysis
- Cross Reference
- Information Flow Analysis
- Data Object Analysis
- MC/DC Test Case Planner
- Generate Instrumented Program(s)
- Enable Exact Semantic Analysis
- Build Instrumented Program(s)
- Execute Instrumented Program(s)
- Dynamic Coverage Analysis
- Dynamic Dataset Analysis
- Profile Analysis
- Dynamic Data Flow Coverage

This analysis, we can visit and generate automated programs, automatically, with this we can configure the instrument, the code, we can compress the aggregation, we can compress into a structure, we can start the analysis, of this selected automated instrumentation.
(Refer Slide Time: 23:18)



Instrumentation Configuration

Instrumented Source File Name (8.3 like)

Analysis Verification Mode (no Instrumentation for Coverage)

Place Instrumented Source Files in: [✓ Original source directories] / Testbed sub-directory / User specified directory

Instrumentation Data File

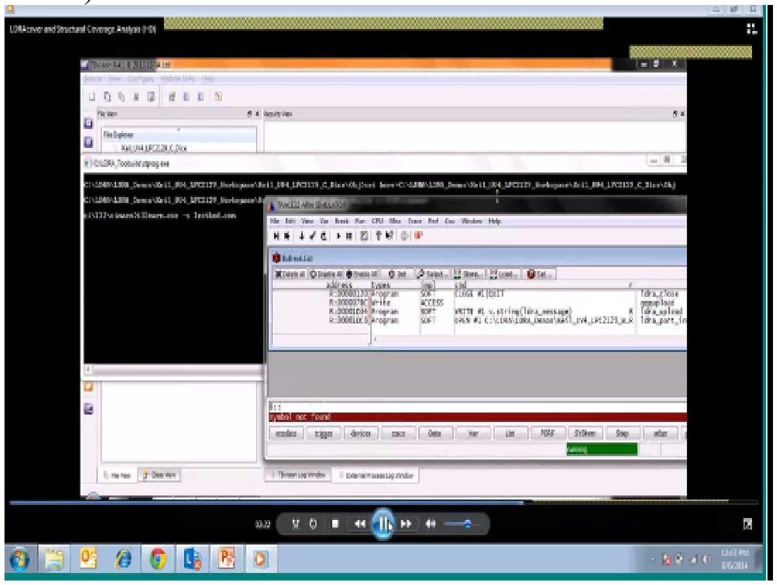
Options (Instrumentation type can only be changed prior to Main Static Analysis)

- [ ] Generate Instrumented File(s) in LRD Format
- [✓] Compressed Execution History
- [✓] Preprocess Instrumented Source File(s)
- [ ] Thread ID in Execution History
- [ ] Use I/O Template (for embedded systems)
- [ ] Execution History to Memory Address
- [✓] Instrument for DO178 DCA
- [ ] Accumulate Bitmap Execution Histories
- [✓] Sustain Execution History in One File

LDRAcover and Structural Coverage Analysis (HD)
Ok
Cancel
Help
TBrun Log Window
External Process Log Window

Let us see, and again .We can say, the target, this instrumented code, it is simply captured, all equations and the structure is going to populate, all the analysis, now we can upload the verification history, and we can analyze the code, which is built already.

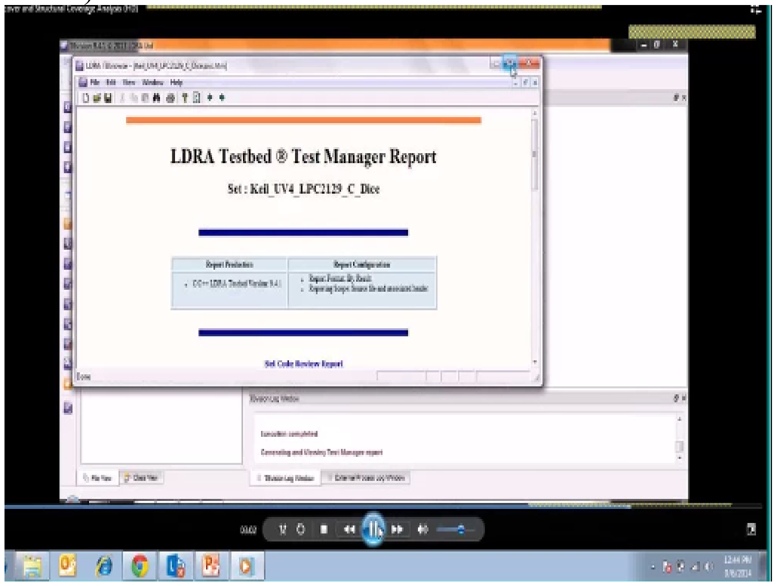
(Refer Slide Time: 24:20)



Okay let us the current trembling execution for that instrumented code, and let us see what is the

result of this, so let us select a project, we need test manager report, and see what does the report Shows now we can see main.C how many times we have executed.

(Refer Slide Time: 25:15)

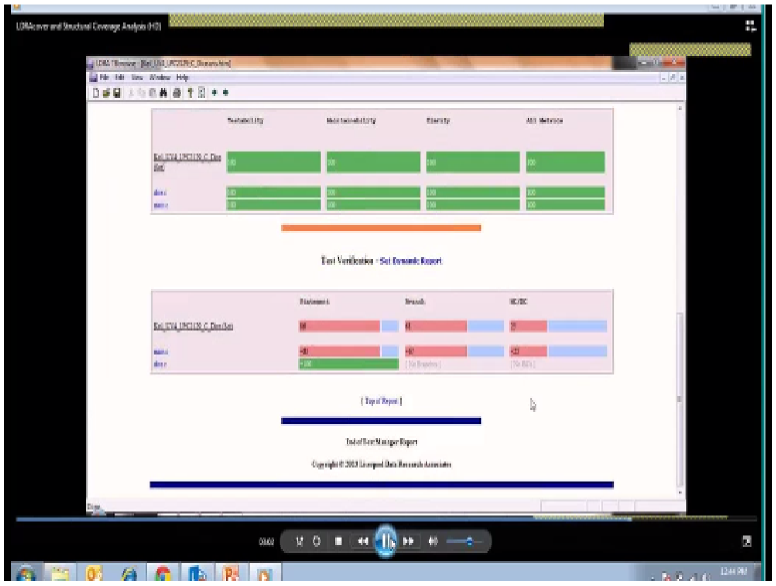

LDRA Testbed ® Test Manager Report

Set : Keil_UV4_LPC2129_C_Dice

Integer to ASCII every line of code how many times you can see here one time executed these

lines.

(Refer Slide Time: 25:28)



	Testability	Maintainability	Clarity	All Metrics
Keil_STVA_LPC2129_C_Dice (list)	100	100	100	100
dice.c	100	100	100	100
main.c	100	100	100	100

Test Verification - Set Dynamic Report

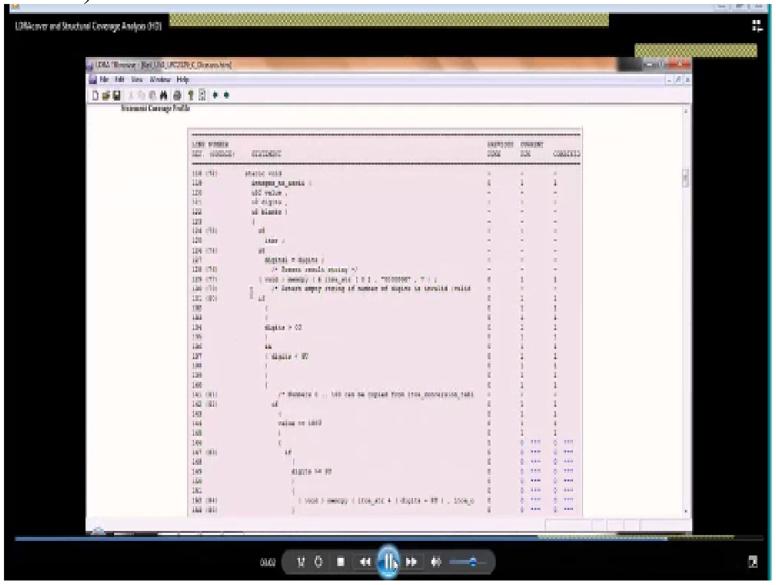
	Statement	Branch	MC/DC
Keil_STVA_LPC2129_C_Dice (list)	[illegible]	[illegible]	[illegible]
main.c	[illegible]	[illegible]	[illegible]
dice.c	[illegible]	[No Branches]	[No MC/DC]

[Top of Report]

End of Test Manager Report

Copyright © 2013 Liverpool Data Research Associates

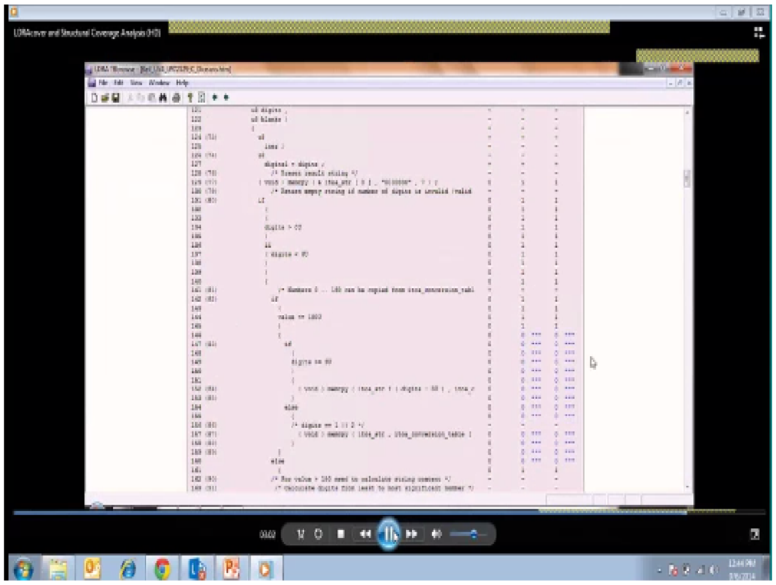
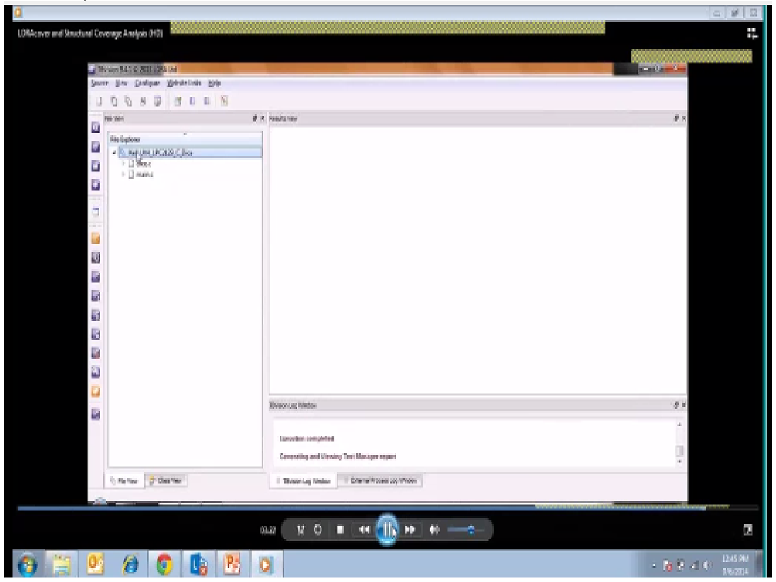
Remains of code here for the instrumentation
(Refer Slide Time: 25:32)

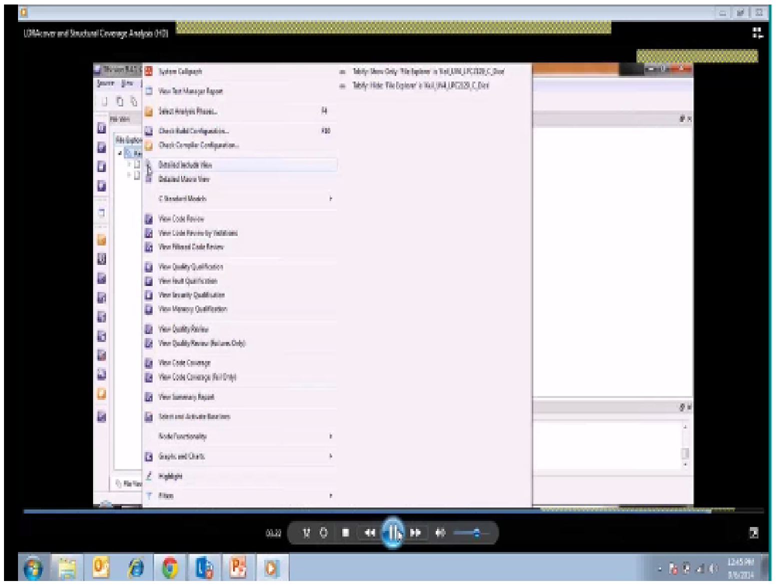


Statement Coverage Profile

```
static void
[illegible]
u32 value ,
u8 digits ,
u8 blanks )
{
  u8
    [illegible] ;
  u8
    digits1 = digits ;
    /* [illegible] */
  ( void ) memcpy ( [illegible] ) ;
    /* [illegible] */
  if
    (
    (
    digits > [illegible]
    )
    &&
    ( digits < [illegible]
    )
    )
    {
      /* [illegible] */
      if
        (
        value <= 1000
        )
        {
          if
            (
            digits >= [illegible]
            )
            {
              ( void ) memcpy ( [illegible] ) ;
            }
```

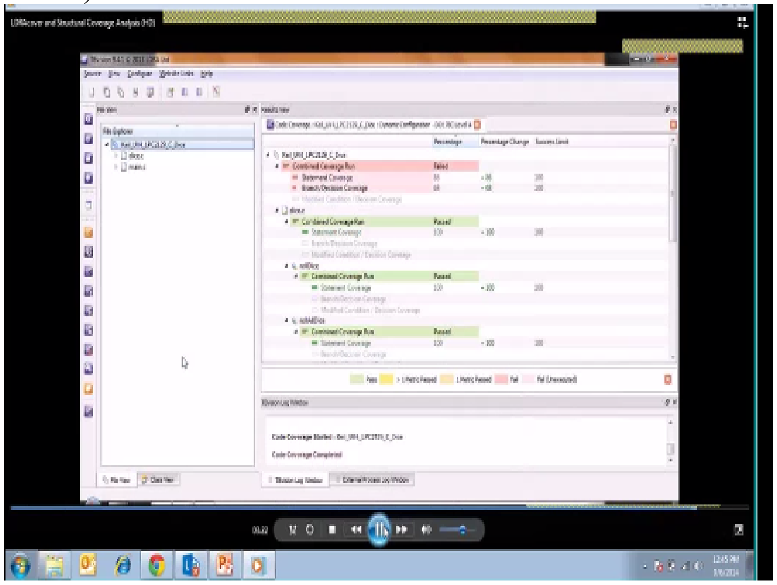
We want to execute this again because we want to have coverage.
(Refer Slide Time: 25:43)


We can see were code overview.
(Refer Slide Time: 25:51)

With that option
(Refer Slide Time: 25:53)



You can see the several overviews we know that entire project has.

(Refer Slide Time: 26:01)



Showing as failed because it is short 49% in terms of statement and we brand the 68% and if you not able to cover with instrumental or raw instrumental then that 49% or 32% of validation you have to do manually or some way of doing it with the other approaches that is what a way to do for the LDRA so basically.

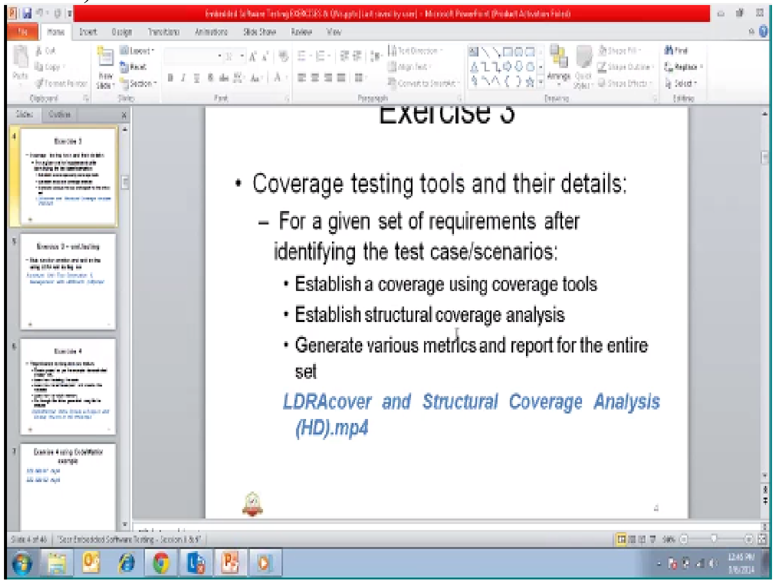
(Refer Slide Time: 26:42)

For further information:

www.ldra.com info@ldra.com

@ldra_technology LDRA Software Technology LDRA Limited

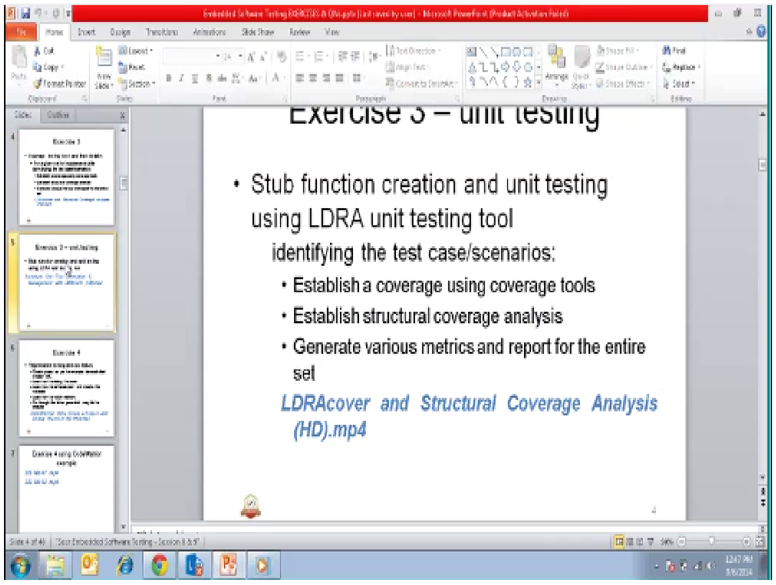
Delivering Software Quality and Security through Test, Analysis & Requirements Traceability

What we have to do is for coverage testing we need to have a source.
(Refer Slide Time: 26:45)

Exercise 3

- Coverage testing tools and their details:
  - For a given set of requirements after identifying the test case/scenarios:
    - Establish a coverage using coverage tools
    - Establish structural coverage analysis
    - Generate various metrics and report for the entire set

  LDRAcover and Structural Coverage Analysis (HD).mp4

We need to have a script and appropriate selection to the LDRA2 and executed and we need to established the coverage using coverage tools then structural coverage analysis of the entire project like this and generate the difference metrics in the proper statement machine coverage and report the entire set is what we have done, the another demo we have stubbed a function at the testing with the help of LDRA unit test, okay with that we will end a today session of the practical thing about Exercise 3 in the next practical session we will take up the next one and a demos at the next practical demonstration.
(Refer Slide Time: 27:46)



Exercise 3 – unit testing

- Stub function creation and unit testing using LDRA unit testing tool
  - identifying the test case/scenarios:
    - Establish a coverage using coverage tools
    - Establish structural coverage analysis
    - Generate various metrics and report for the entire set

    LDRAcover and Structural Coverage Analysis (HD).mp4

4