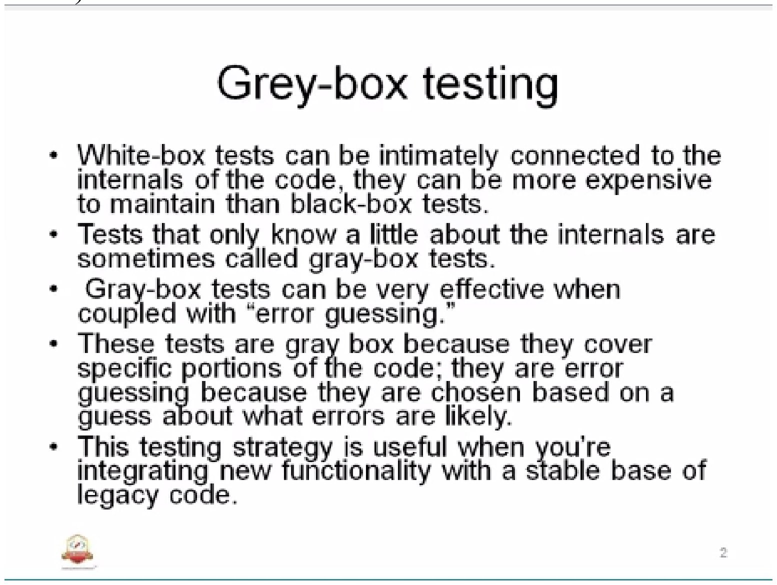
Embedded Software Testing
Unit 2: Testing Methods
Lecture 22
Seer Akademi-NPTEL MOU

Embedded software testing, this will be the last session of unit 2 and we will study about some of the white box testing techniques and understanding of testing aspects.

(Refer Slide Time 00:23)



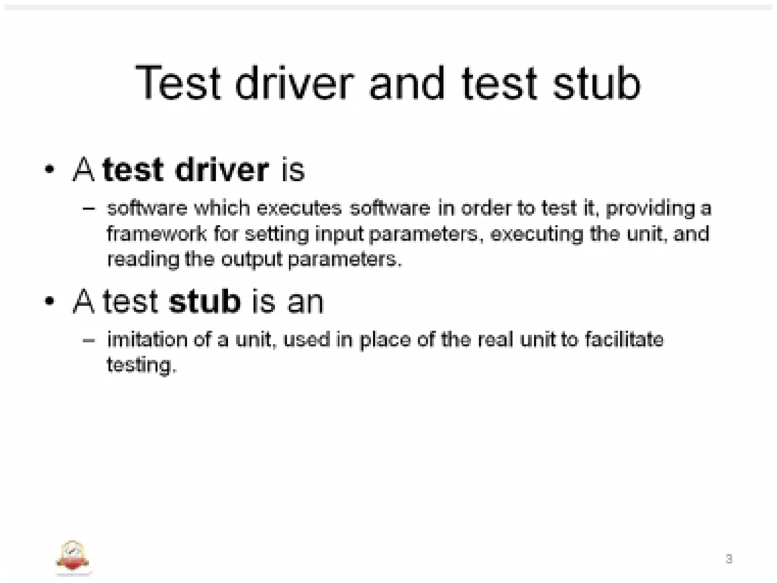
Grey-box testing

- White-box tests can be intimately connected to the internals of the code, they can be more expensive to maintain than black-box tests.
- Tests that only know a little about the internals are sometimes called gray-box tests.
- Gray-box tests can be very effective when coupled with "error guessing."
- These tests are gray box because they cover specific portions of the code; they are error guessing because they are chosen based on a guess about what errors are likely.
- This testing strategy is useful when you're integrating new functionality with a stable base of legacy code.

2

In the previous session we study about grey box testing which is nothing but a fix of white box as well as black box test where tester as the knowledge about internal details of the system as well as system knowledge with the help of that and we will do the testing. This is actually very effective when it is coupled with error guessing where, with the help of the system knowledge he will guess what sort of errors will come it is particularly useful when we have a stable system working. The legacy code and there are certain changes or new functionality has been added to the impunity system.

(Refer Slide Time 01:11)

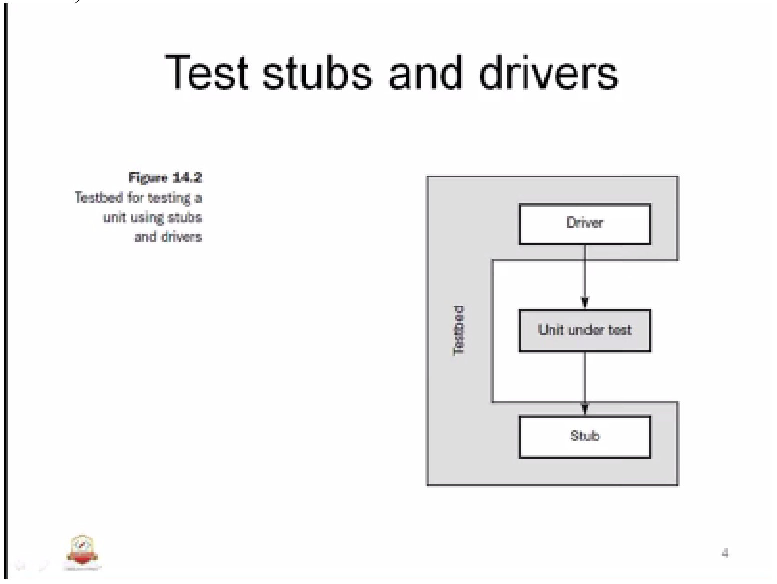


Test driver and test stub

- A **test driver** is
  - software which executes software in order to test it, providing a framework for setting input parameters, executing the unit, and reading the output parameters.
- A test **stub** is an
  - imitation of a unit, used in place of the real unit to facilitate testing.

3

Also we study about test driver and test stub, which is the important part of the embedded software testing. Test diver is software which executes software in order to test it, providing a framework for setting inputs parameter, executing the unit, and reading the output parameters. So basically test driver will invoke the test which want to test it, there is also a software piece with whatever the values that we want to provide as inputs and expect the outputs. Test stub is the replacement of the exiting unit, so this is used in place of the real unit which is facilitate testing.
(Refer Slide Time 01:47)

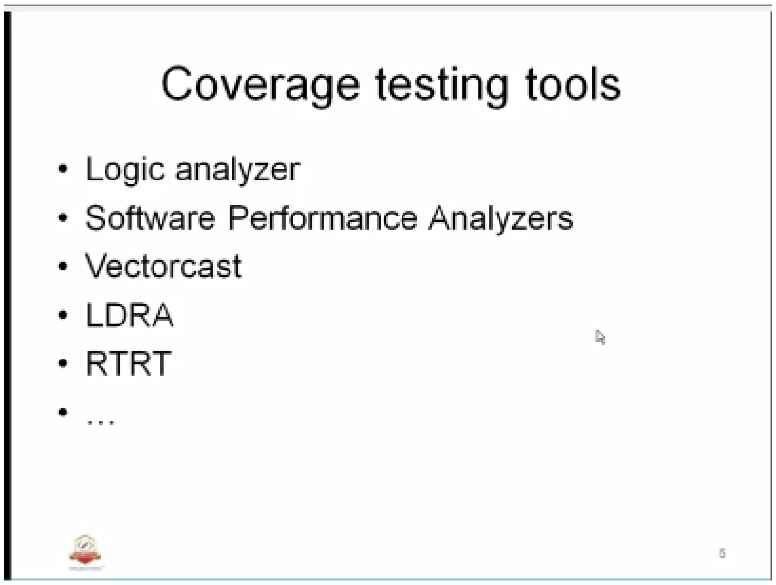


Test stubs and drivers

Figure 14.2
Testbed for testing a unit using stubs and drivers

4

So this is the train work that is being used for test stub and test driver, this is called a test stub job. Basically so the impunity is get used by the driver, test driver and unit can be replaced with the help of a test stub. So test stub and driver are part of the testbed the unit will be subjected to test.
(Refer Slide Time 02:11)

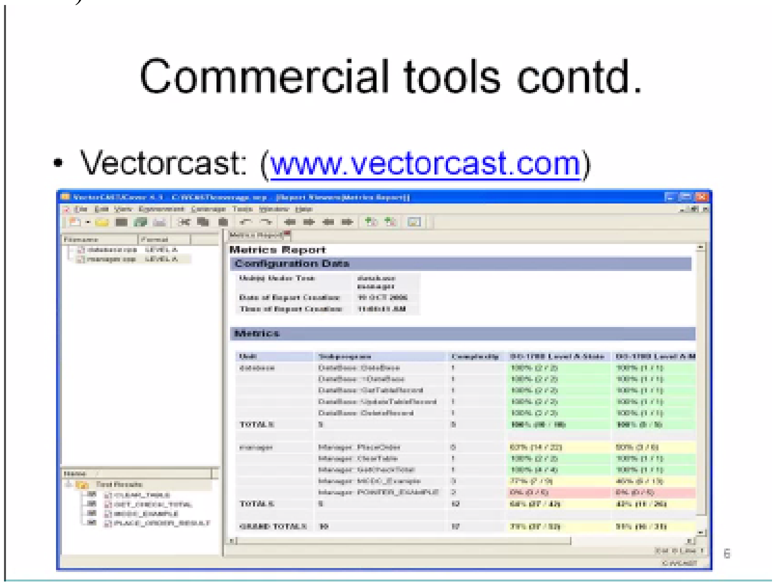


Coverage testing tools

- Logic analyzer
- Software Performance Analyzers
- Vectorcast
- LDRA
- RTRT
- …

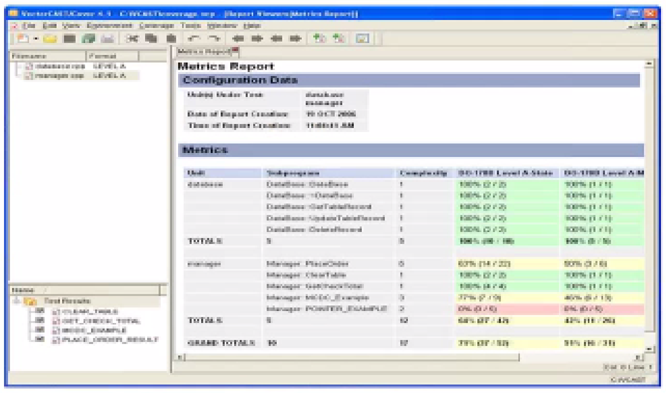
5

Coming to the various aspects of the tools that are used in white box testing, basically they are used for coverage purposes and we have study about commercial tools such as vectorcast
(Refer Slide Time 02:27)



Commercial tools contd.

- Vectorcast: (www.vectorcast.com)



Metrics Report

Configuration Data

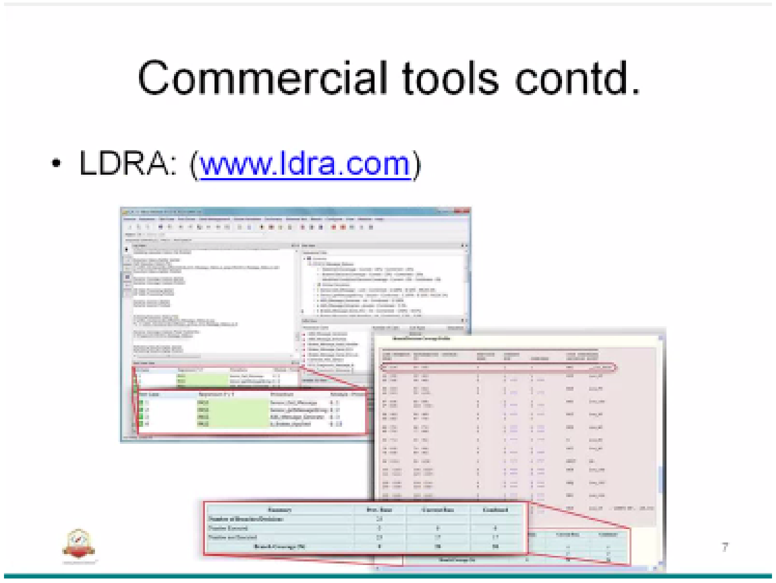
Unit(s) Under Test	database manager
Date of Report Creation	19 OCT 2006
Time of Report Creation	[illegible] AM

Metrics

Unit	Subprogram	Complexity	DO-178B Level A-State	DO-178B Level A-M
database	DataBase::DataBase	1	100% (2 / 2)	100% (1 / 1)
	DataBase::~DataBase	1	100% (2 / 2)	100% (1 / 1)
	DataBase::GetTableRecord	1	100% (2 / 2)	100% (1 / 1)
	DataBase::UpdateTableRecord	1	100% (2 / 2)	100% (1 / 1)
	DataBase::DeleteRecord	1	100% (2 / 2)	100% (1 / 1)
TOTALS	5	5	100% (10 / 10)	100% (5 / 5)
manager	Manager::PlaceOrder	5	63% (14 / 22)	50% (3 / 6)
	Manager::ClearTable	1	100% (2 / 2)	100% (1 / 1)
	Manager::GetCheckTotal	1	100% (4 / 4)	100% (1 / 1)
	Manager::MCDC_Example	3	77% (7 / 9)	46% (6 / 13)
	Manager::POINTER_EXAMPLE	2	0% (0 / 5)	0% (0 / 5)
TOTALS	5	12	64% (27 / 42)	42% (11 / 26)
GRAND TOTALS	10	17	71% (37 / 52)	51% (16 / 31)

6

For cover for getting the reports of the coverage and we also went through,
(Refer Slide Time 02:33)



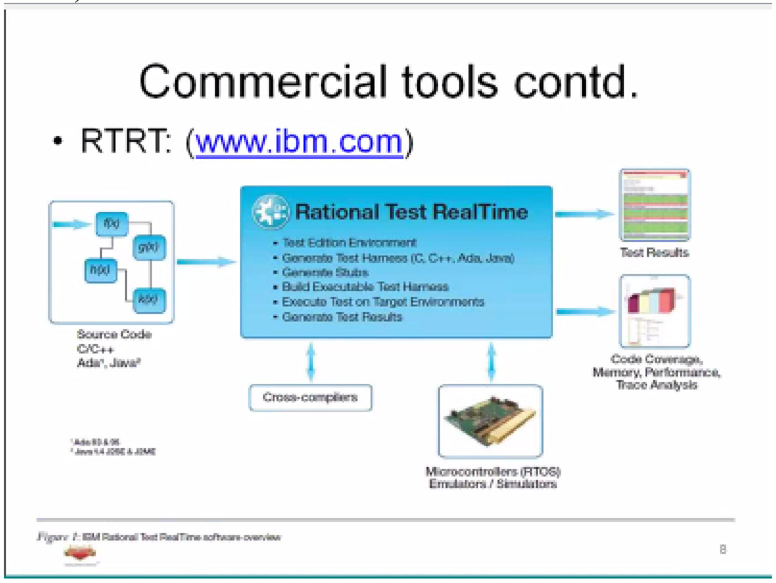
Commercial tools contd.

- LDRA: (www.ldra.com)

7

LDRA from ldra.com and IBM, RTRT, so which are specialty for doing the test stubs building the executing above from the targets and generating the reports.

(Refer Slide Time 02:38)



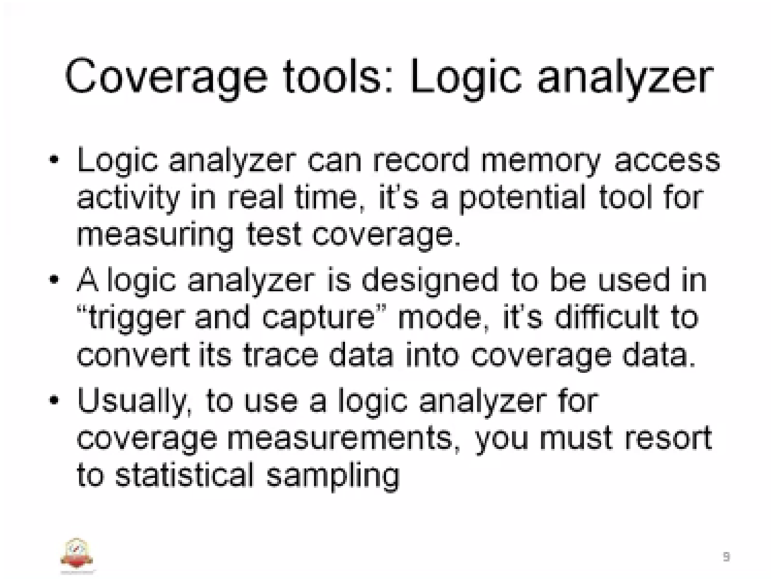
Commercial tools contd.

- RTRT: (www.ibm.com)

Figure 1: IBM Rational Test RealTime software overview

8

(Refer Slide Time 02:38)



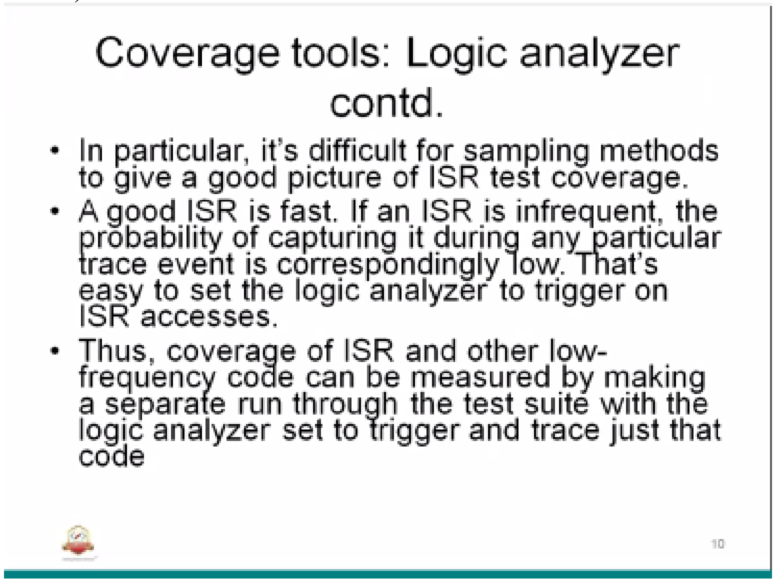
Coverage tools: Logic analyzer

- Logic analyzer can record memory access activity in real time, it's a potential tool for measuring test coverage.
- A logic analyzer is designed to be used in "trigger and capture" mode, it's difficult to convert its trace data into coverage data.
- Usually, to use a logic analyzer for coverage measurements, you must resort to statistical sampling

9

Also we studied about logic analyzer which will help in recording the memory activity in a run ventime or run time and records the switch changes and updates that are in the embedded software. Basically captures the events that are happening and this is called the trace. This is particularly useful for inter based application

(Refer Slide Time 03:18)



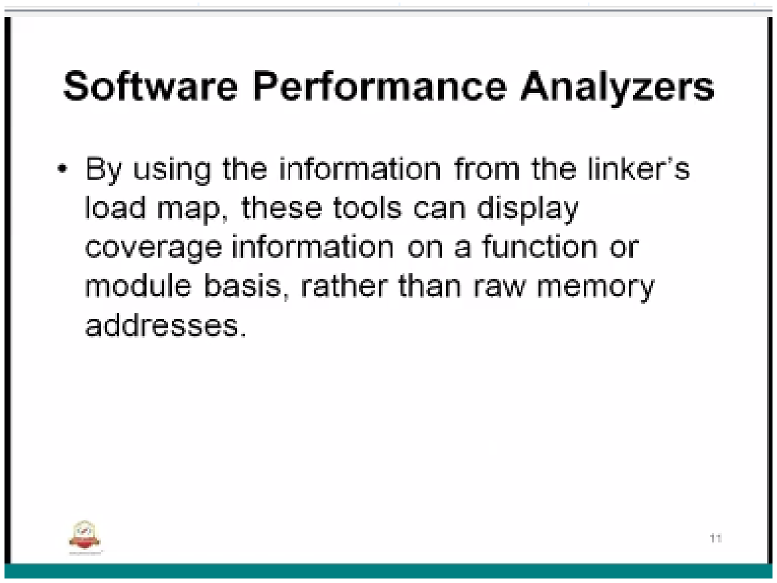
Coverage tools: Logic analyzer contd.

- In particular, it's difficult for sampling methods to give a good picture of ISR test coverage.
- A good ISR is fast. If an ISR is infrequent, the probability of capturing it during any particular trace event is correspondingly low. That's easy to set the logic analyzer to trigger on ISR accesses.
- Thus, coverage of ISR and other low-frequency code can be measured by making a separate run through the test suite with the logic analyzer set to trigger and trace just that code

10

Where more interrupts are there and applicant of the interrupts are more.

(Refer Slide Time 03:26)



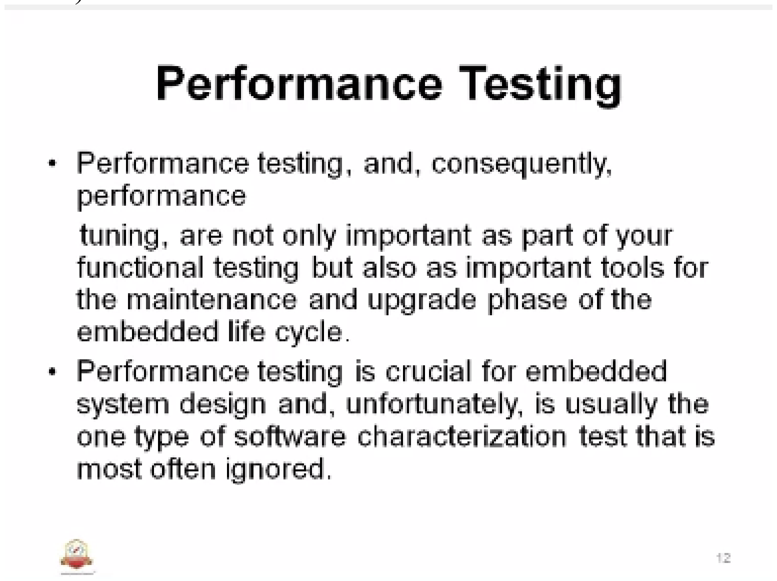
Software Performance Analyzers

- By using the information from the linker's load map, these tools can display coverage information on a function or module basis, rather than raw memory addresses.

11

Also we had study about software performance analyzers, the performance could be from memory, timing, and load, speed, etc,

(Refer Slide Time 03:45)



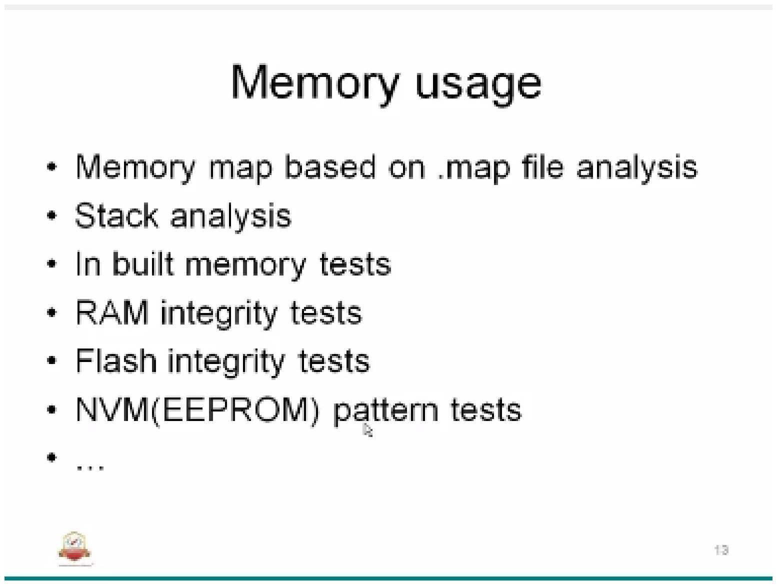
Performance Testing

- Performance testing, and, consequently, performance tuning, are not only important as part of your functional testing but also as important tools for the maintenance and upgrade phase of the embedded life cycle.
- Performance testing is crucial for embedded system design and, unfortunately, is usually the one type of software characterization test that is most often ignored.

12

So performance testing and retuning is very important aspect because that will try the functionality and especially if the software as to be upgradable or maintainable. So the performance as to be absolutely right, then only we can do the rest of the testing so this is very important aspect sometimes in the industry this will be taken as a last priority but as for the main it should be a top most priority for doing the embedded software testing.

(Refer Slide Time 04:18)

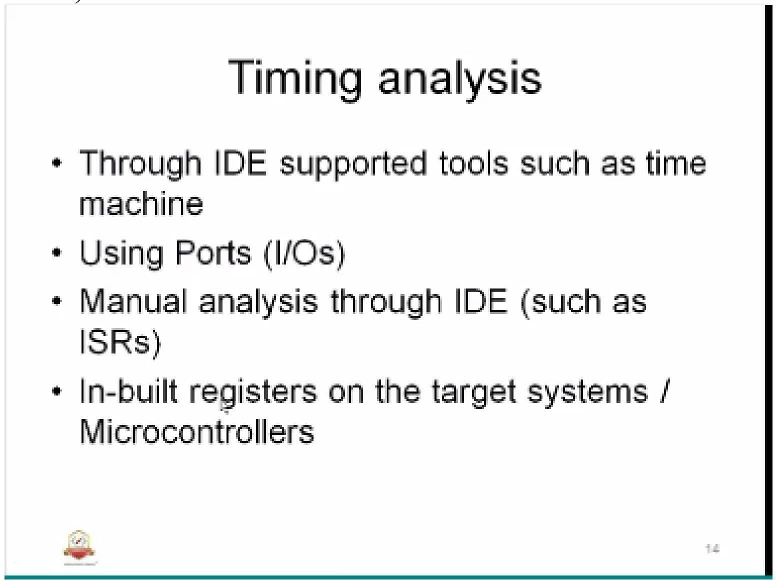


Memory usage

- Memory map based on .map file analysis
- Stack analysis
- In built memory tests
- RAM integrity tests
- Flash integrity tests
- NVM(EEPROM) pattern tests
- ...

13

Also there are various types of memory testing in the, so especially they look for the memory usage all the memory is being used in the embedded software system. And there is a map file, and the map file will be detailed in terms of how much stack, how much constant are using for the size of the RAM, what is the size of the clash etc,. And for testing each of this memory parts in the embedded software, basically the software itself and it will be keep on checking in every fixed do intervals or cycles of the system life. And the reports will be generated or cooked with the additional to get the data and that will be analyzed. So basically the RAM, memory, flash will be checked with the help of integrity test, integrity test could be working once or a pattern test (NVM) also will be tested with the help of pattern test or the working test. Where (EEPROM) are used on the fault of recording the storage.
(Refer Slide Time 05:30)



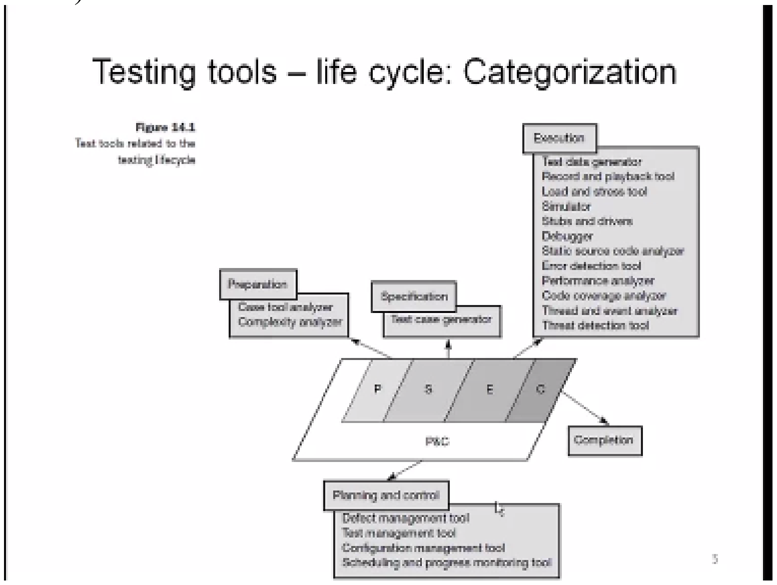
Timing analysis

- Through IDE supported tools such as time machine
- Using Ports (I/Os)
- Manual analysis through IDE (such as ISRs)
- In-built registers on the target systems / Microcontrollers

14

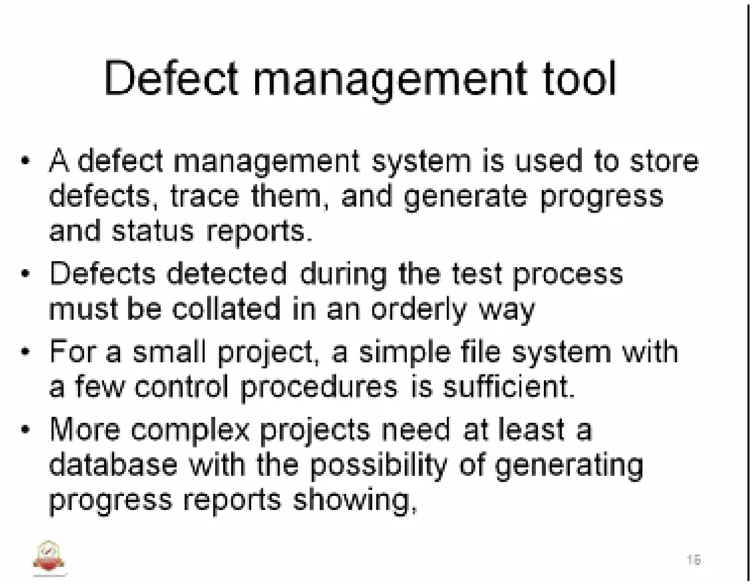
And timing analysis we can do with the help of the integrity development and (NR) mind, with the help of debuggers we can analyze the inputs of reporting. And the various events that are happening with respective time, and as a black box so it want to test it, we may have to use codes if it is available with the embedded system in those parts can be scoped in the oscilloscope and oscilloscope will be analyzed with respective the plotted graph of timing with respective events. Also the embedded system as hipsters with the respective time such as RTC, watched all, etc, those will be used as well in terms of analyzing time.

(Refer Slide Time 06:14)



Also we went through the testing tools life cycle, how the life cycle can be used with respective tools. So we have planning and control of the entire testing life cycle which as preparations, specification, application and complication. So preparation we have tools such as case tool analyzer, complexity analyzer. Specification we have test case generator of the test case generator it as got a lot of pipe and scripts or excel sheet programming from the requirements and so and so we have execution in terms of test case generator, simulators, debuggers, those are we studied so far one of them are many of them could be used for execution purpose. And we have the completion and reporting mechanism of the executed parts for planning and control we use defect management and test management, interstablity etc, with respective to the requirements to the testing.

(Refer Slide Time 07:29)



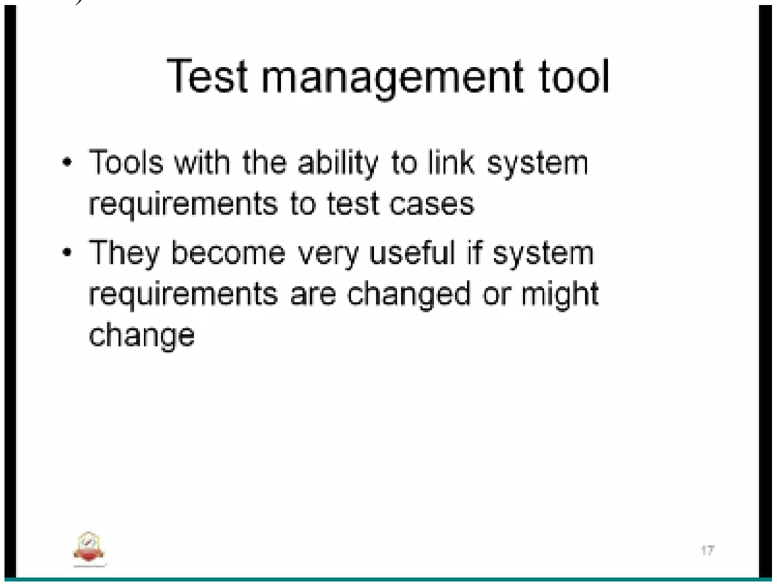
Defect management tool

- A defect management system is used to store defects, trace them, and generate progress and status reports.
- Defects detected during the test process must be collated in an orderly way
- For a small project, a simple file system with a few control procedures is sufficient.
- More complex projects need at least a database with the possibility of generating progress reports showing,

16

Defect management we have studied depending in the size of the part size of the project where types of the defect complexity and all that will be managed.

(Refer Slide Time 07:37)



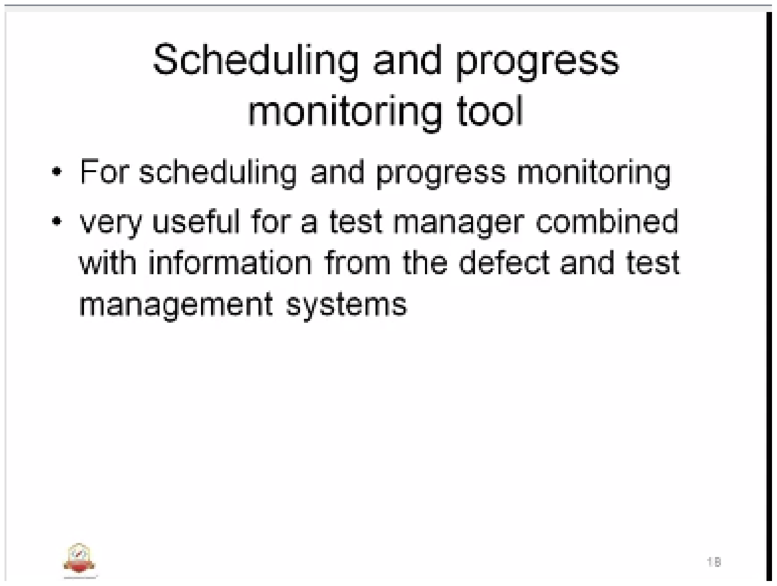
Test management tool

- Tools with the ability to link system requirements to test cases
- They become very useful if system requirements are changed or might change

17

Is the data base and all that they will be used in order to get the report of the defect as well as the requirements testability?

(Refer Slide Time 07:48)



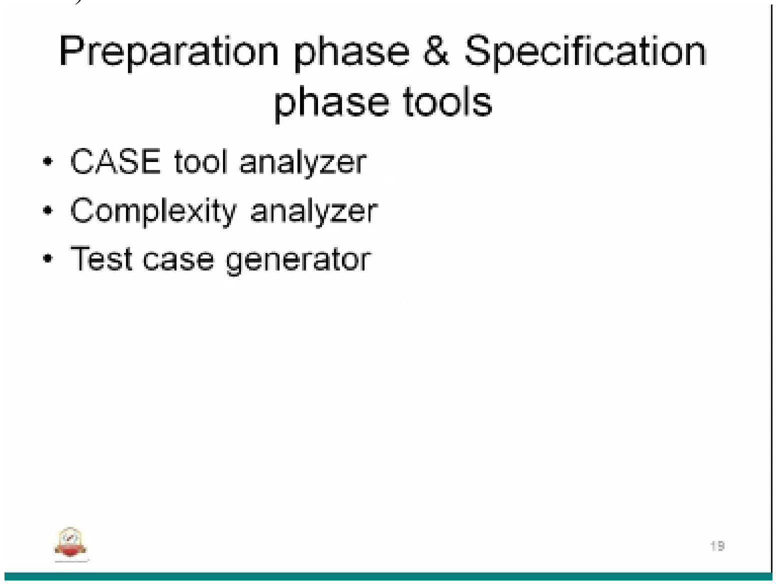
Scheduling and progress monitoring tool

- For scheduling and progress monitoring
- very useful for a test manager combined with information from the defect and test management systems

18

And in terms of scheduling and are used, different types of tools such as MPP and test links bugzilla whatever it is can be used in addition to scheduling. This will help test management and progression and the embedded software testing.

(Refer Slide Time 08:12)



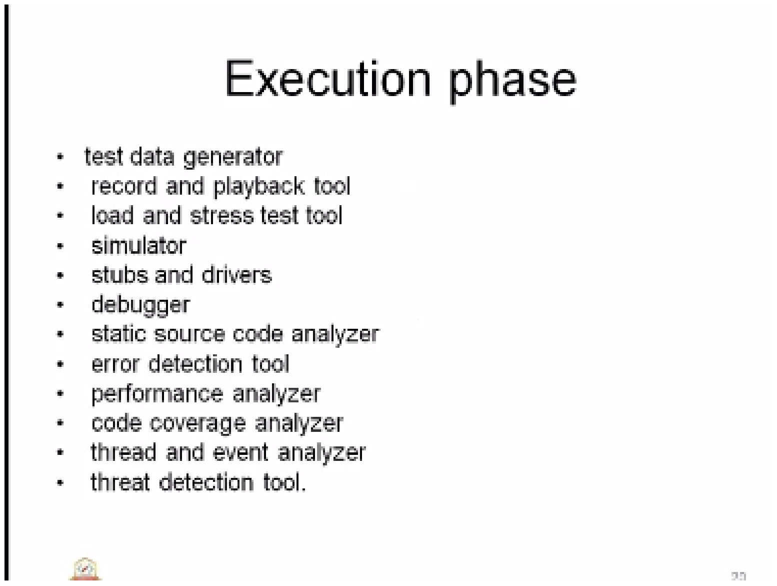
Preparation phase & Specification phase tools

- CASE tool analyzer
- Complexity analyzer
- Test case generator

19

Test case preparation, specification are when the help of case tool analyzer, complexity analyzer, test case generators for the specification they use it.

(Refer Slide Time 08:23)



Execution phase

- test data generator
- record and playback tool
- load and stress test tool
- simulator
- stubs and drivers
- debugger
- static source code analyzer
- error detection tool
- performance analyzer
- code coverage analyzer
- thread and event analyzer
- threat detection tool.

Execution we have seen stubs and drivers, simulators, emulator, debuggers, static code analyzers, performance analyzers, timing analyzers, code coverage analyzers are used these are all basically used in white box testing as well as black box testing.

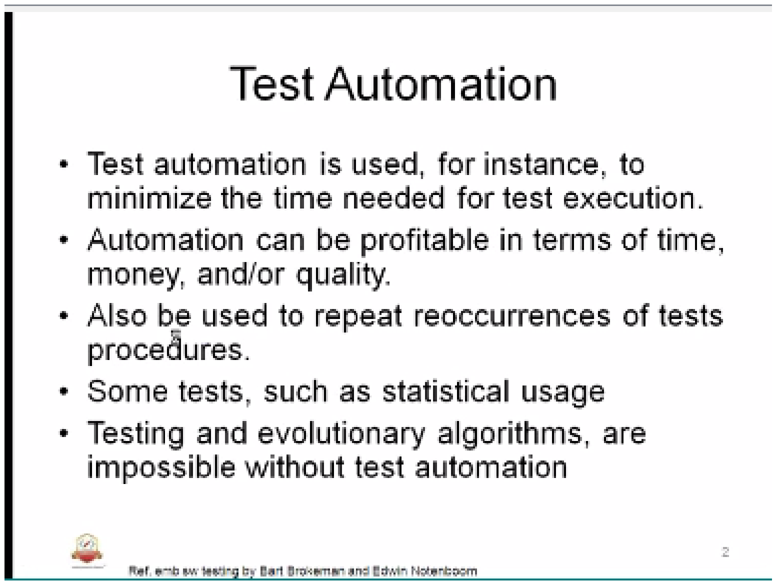
(Refer Slide Time 08:49)

Embedded Software Testing
Unit 2: Testing Methods

Lecture 22
Seer Akademi – NPTEL MOU

1

(Refer Slide Time 08:53)



Test Automation

- Test automation is used, for instance, to minimize the time needed for test execution.
- Automation can be profitable in terms of time, money, and/or quality.
- Also be used to repeat reoccurrences of tests procedures.
- Some tests, such as statistical usage
- Testing and evolutionary algorithms, are impossible without test automation

Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

2

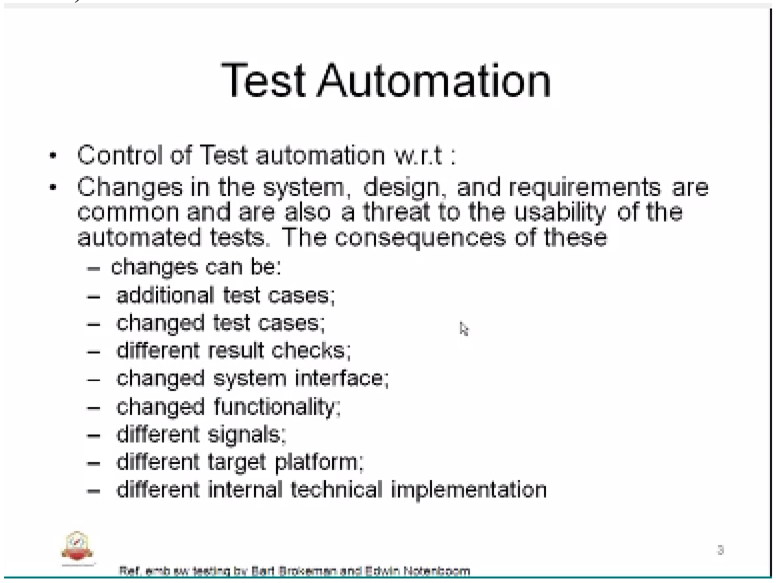
And today's session we will study more about the white box method how they are getting implemented? And how they can be executive? And what way we can automicate. So basically test automation is process or a test method or whatever the testing aspects that can be applied for testing the embedded software. So basically test automation is used to minimize the time lead for test execution. So why we need test automation is we will keep on doing the embedded software testing again and again and similar requirements, similar sort of test cases, so we need to have a common framer and common framer we will drive the entire system to test.

And we will feed the values with respective each test cases, and that is why the test framer along with the execution test automation is useful. So automation is very profitable in terms if time money and quality, but it needs a initial needs investment of time one time, once it done the framer keys is available it is very easier to maintain as well as run the test automatically.

So it is very less human intention is required in terms of testing, and also it can be se to repeat reoccurrences of tests procedures, that's means so multiple test of similar procedures we don't need to built manually we can do a batch processing batch automation it is also called as, so test automation will help in during these things, there are multiple procedures are to be executed in a batch, at some test such as statistical usage also can be done with the help of test automation, testing and evolutionary algorithms are impossible without test automation that means there are certain things which are repetitive nature or algorithms which keep changing we have a set of data and there are fixes and you may have to do regression.

We cannot offer to have a manual, so in that way it is very useful to have test automation, so basically in principle the execution of almost every test can be automated depending on type of test in practice so only a small part of test are automated in depending on the project again, about the automation will be done most of the requirements where functionality or group of a functionality can be tested with the various inputs or various multiple test cases in executed, so basically test automation is offend used in a different situation where test that have to be repeated many times, the basic test with huge range of inputs are there, the saying steps have to be executed on that with different data, very complicated or error born test in this we have a piece of

software having more errors and all that and every time different errors are coming so it is better to automate it so that minimal human intervention is used for that.
So it is very useful for complicated tests, but testing requires specialized equipment which will generate the appropriate input signals active of the system and captures the output and analyze the output. Analysis can be done manually or a tool itself can be analyzing it.
(Refer Slide Time 13:10)



Test Automation

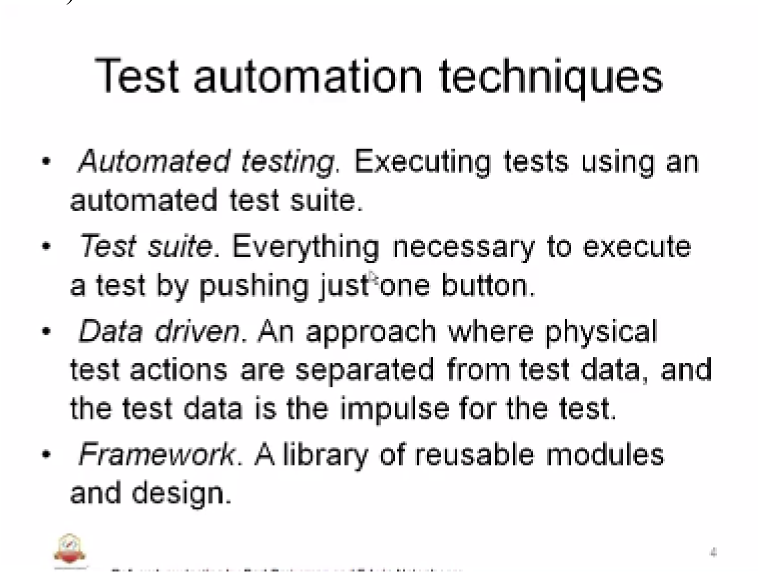
- Control of Test automation w.r.t :
- Changes in the system, design, and requirements are common and are also a threat to the usability of the automated tests. The consequences of these
  - changes can be:
  - additional test cases;
  - changed test cases;
  - different result checks;
  - changed system interface;
  - changed functionality;
  - different signals;
  - different target platform;
  - different internal technical implementation

3

Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

So continuation of test automation so we can have the control of test automation prospective changes in the system, design and requirements are common and are also a threat to the usability of the automated test. So what will happen is we have test automation completely for the entire terms of the embedded software and the requirements. Sometimes the changes are the fixes that could happen to the system or the tests that are going to be applied of the system.
So the consequences of those changes could be changes on test cases, additional test cases are with the different aspects. When different results checks are there and framer as some impact in terms of modifying it, system interface as changed, there is change in functionality so signals have changed, target platform is different, from one built to another built there are different hardware boards with different version is being used.
Then there is a possibility that automation may fail or collapse, so control of the test automation is very important it should have the scalability of the test automation also. Automation equipment or tools or scripts that are developed should be more generic than specific so that it is very maintainable and all the specified consequence of changes will have a less impact on modifying the test automation updates and regression.
So basically automated test saves money and time and quality will be better because less human intervention is there we are depending on the system and tools of the framer. In the framer key is stable once you make it, it will expected to behave in the same way. So chance of injecting error by the user is very less and several times that can test can be reproducible. So this is also very important reproducibility of the test execution pass fails etc., is easier with the help of test automation. So this means that the automated test should be used for consequencative releases are in difference stages of development.

That means the same sort of automation should be able to use for different reason of the software built and the equipments sorry the embedded target. So which also means that automated test have to design to deal with these uncertainties due to change in the system design are whatever it is. So we will see how automation test can be implemented technically in addition, basically we will try to understand that and automation test basically being developed in house with the help of commercial tool basically.
That are available in market and of course those tools have to be full proof and should be qualified especially in the industries of automotive and aerospace it is very important because it as to be reliable and we need to produce a consistence result all the time. So then also we will study about the other process of introducing automation testing and similar development process such as requirement and employment maintenance.
(Refer Slide Time 17:35)

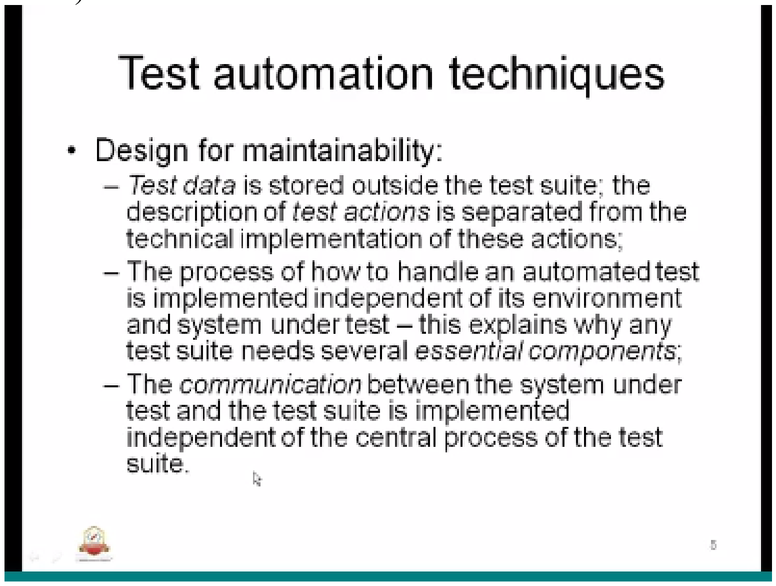


Test automation techniques

- *Automated testing*. Executing tests using an automated test suite.
- *Test suite*. Everything necessary to execute a test by pushing just one button.
- *Data driven*. An approach where physical test actions are separated from test data, and the test data is the impulse for the test.
- *Framework*. A library of reusable modules and design.

So test automation techniques automated testing, executing tests using an automated test suite so there is a test suite that will be used. Which will be executed for testing is specific software piece. Test suite, everything necessary to execute a test by pushing just one button ideal that should be that actually, test suite will be triggered and it will take care of all controlling, picking the data signals and driving the values and getting the results analyzing it reporting as pass or fail is all part of the test suite.
Then test automation should be data driven, an approach where physical test actions are separated from test data, and the test data is the impulse for the test. This will be data driven and this should be a frame work in the end a library of reusable modules and design as you see test stubs and drivers there should be reusable for many parts of the system. So test stubs, test drivers, then common templates, or it could be a minimum sequences in especially I have seen lab view and simui link it is very important the sequences the minimum templates the develop it and they can use it as a plug and play for wherever they want to use within the complete automated testing system. So there is a test suite, test suite will use this frame work for the need of that particular test. So to maximize maintainability, possibility and use of this, so some

decision denied decision have to be made. To avoid a test tool from becoming itself where it is designed to be as independent as possible of changes so that the frame work can be used.
So it is very important to have a test suite and the system connectivity and test that are driven also is very important in terms of identifying the approach for the underneath frame work. So generically the frame work is developed to avoid the huge dependence basically. So where the dependence is there we will try to minimize as much as possible so that the frame work is generic and the framer can be used across multiply functionality of the pieces of the software under the test.
(Refer Slide Time 21:07)



Test automation techniques

- Design for maintainability:
  - *Test data* is stored outside the test suite; the description of *test actions* is separated from the technical implementation of these actions;
  - The process of how to handle an automated test is implemented independent of its environment and system under test – this explains why any test suite needs several *essential components*;
  - The *communication* between the system under test and the test suite is implemented independent of the central process of the test suite.

5

So design for maintainability test data is stored in continuation of test automation techniques so we had a test suite, data driven, and frame work, then for maintainability purposes test data is stored outside the test suite the description of the test action is separated from the technical implementation of these actions. I will show you a diagram which is easy to understand with the help of that you can understand these descriptions.
A process of how to handle an automated test is implemented independence of its environment and system under test; this explains why any test suite needs several essential components, essential components could be a test data, and the frame work and the test actions inbuilt in the test suite, the communication between the system under the test and the test suite is implemented independent of the central process of the test suite.
(Refer Slide Time: 22:07)

Test automation techniques ex.

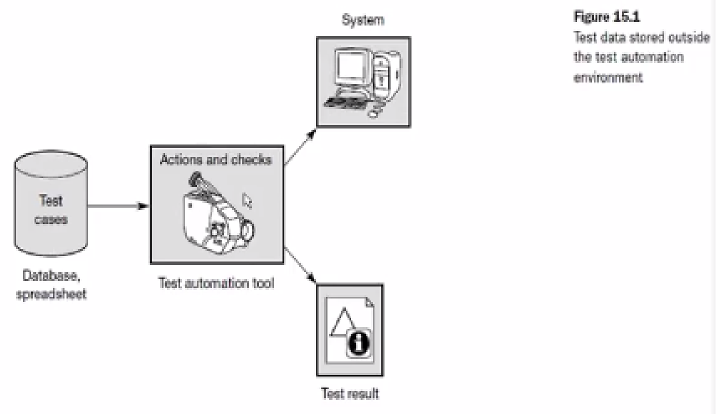
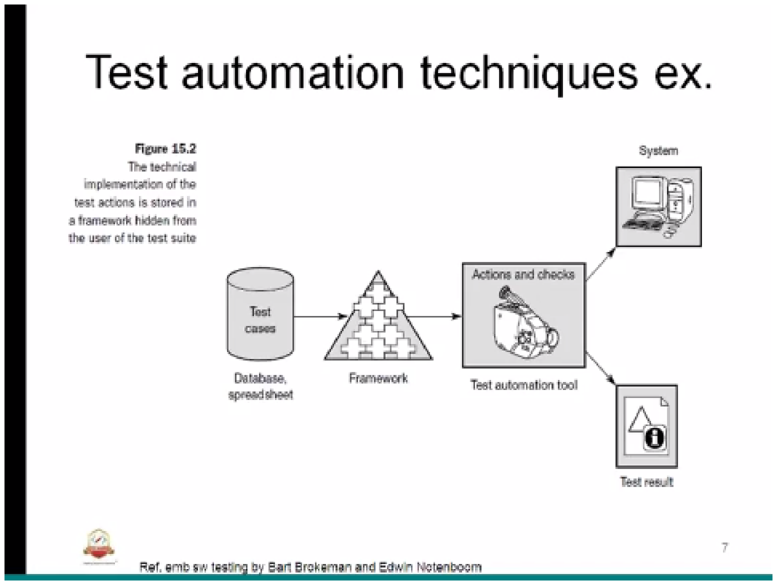


Figure 15.1
Test data stored outside the test automation environment

So we can see an example on test automation and environment how it is going to look like, so it is just a depiction. So it does not mean that same thing is used in action the system that is going to be tested is here in the system and we have a repository of test cases. So all the test cases will have the data base, spread sheet whatever it is that will be picked by the automation two, it will pick up that and it will run on the system and it will generate the result.
Basically these are flow where all the actions checks comparison all be done with the help of this test automation tool. And the data is stored outside all the data test cases those things so you know that the framer is part of this, this framer could pick up the test cases and it will apply the data bases as per the inputs. And it will execute in the target here and as a result of that it will generate the report and the outcome of the test.
In a spread sheet or excel sheet or text document or the logging whatever is required, of course also it will use the facility for the inter cases of the embedded systems such as debuggers and all that everything can be automated with the help of the automation tool. Even the invoking of ID, debuggers, analyzers, or scope also can be treated with the help of this automation all these we need you know that. So if you automate it is easier for us to maintain and designing the test aspect so that is how testing is done this is particularly useful where we have a complex for embedded software system.
So on where the variant of, so test case are stored outside the test suite they can store the data base in the spread sheet here and variant test case means the test cases to the storage of the test case. So there is a in additional test case what we do you do not have to change anything on this side only you have to add the depository or delete or modify whatever it is so the test suite do not have to change.
(Refer Slide Time: 25:25)



Test automation techniques ex.

Figure 15.2 The technical implementation of the test actions is stored in a framework hidden from the user of the test suite

7

Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

So the test suite will all ways be same, so another variant of this can see in the next diagram where test actions and checks as part of the test automation tool will work like a machine and frame work is outside. Frame work is built little outside with more automation here, and the philosophy of test case, data base, spread sheet are all on the left hand side in the depository so the technical implementation of the test actions.

That are part of this are stored in the frame work we can see the blocks these blocks can be a serial of action, the sequences so that is return from the user of the test suite. So this is a test suite basically again the test suite will have a system trigger and the specific test cases are picked up and the frame work will drive what needs to be done. So that is difference between this and S1, where we have the frame work separated from the automation tool.

Of course we can also call it as an independent automation tool, this bundle is very important here where maximum automation is done. With the help of frame work and the action specific tool, this tool could be lab view test and or some link or the multiply tools available in the market. So basically the technical implementation of certain test action can be rather complicated so this implementation should be hidden from the normal user.

We need not to bother about where it is available, so only thing to know is which test action are available so those action are part of the automation. So automation pick from the frame work so if the functionality of the system retest space the same but the technical implementation is changed suppose you have different implantation all together on the embedded systems the frame work will modify, so you have to modify the automation itself, so in that aspect to be useful, for example an addition inter faces is added or signals have change in the character tics etc…. but user can still use the same script if nothing has happened. So in the previous example we have seen that test data and actions are combined in one test script, whereas here we have separate out with the help of the frame work, so that is the thing.

(Refer Slide Time 28:49)

Test automation techniques: Test suite

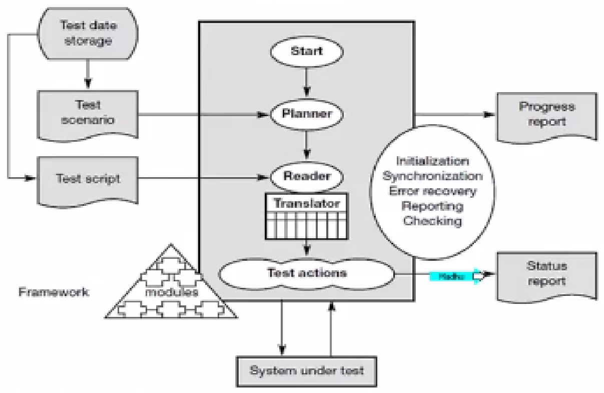


Figure 15.3 Blueprint of a test suite

Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

Next one is the test suite how its looks like the inter cache of the test suite basically it is also called a in the book it is specified by map of the testing by Bart Brokeman and Edwin Notenboom.

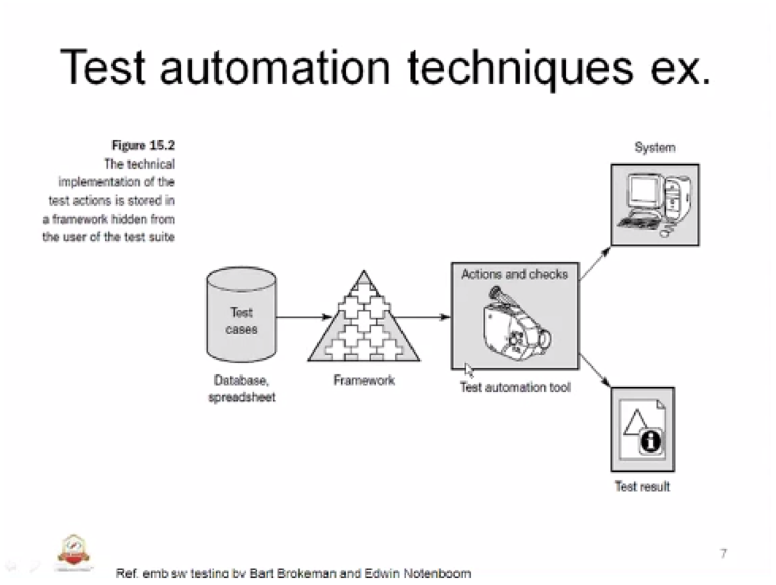
Am referring most of the pieces there because that is more relevant what I see you can refer the physics book online, so test suite has several components inbuilt, so lets us see how to close, so this the basic what is that called frame work having different modules you can see it here and it has a flow, this is a start, this is the plan work, this is the reader, the planner will pick up particular test scenario on the test data storage, that means we have a test data storage in terms of a data base having signals its definition and all that and the cases also.

Test cases also specifying the test scenario and that scenario will picked by the planner module, and the next one will be reader so the particular scenario, the particular test script will read from the stored data base or the Contents.

And there is a translator, the transector could be applying the script in to the understandable actions and to be target, so the test actions are derived from the translators as the intermediate layer, so the name is translator but it could be anything when it is comes to the actual embedded software testing automation in different name could be used. So the actions could be initialization, synchronization, error recovery, reporting, checking and all that part of this frame work having different modules, and as the result of test actions we have the status required and the actions will be applied on the target system, system and the test and the status report will tell that for the each of the script and the scenario what is the pass fail value, or the what are the passes and where it has failed and also when we do the batch alone we can get the progress report in terms of pass fail count. It is called pass fail count bound own progress is basically a chart of the entire test suit or the bundle of the test suit progress so, that is how the suit in terms of adoration.

So, this is about the suit so where the,

(Refer Slide Time 31:44)



Test automation techniques ex.

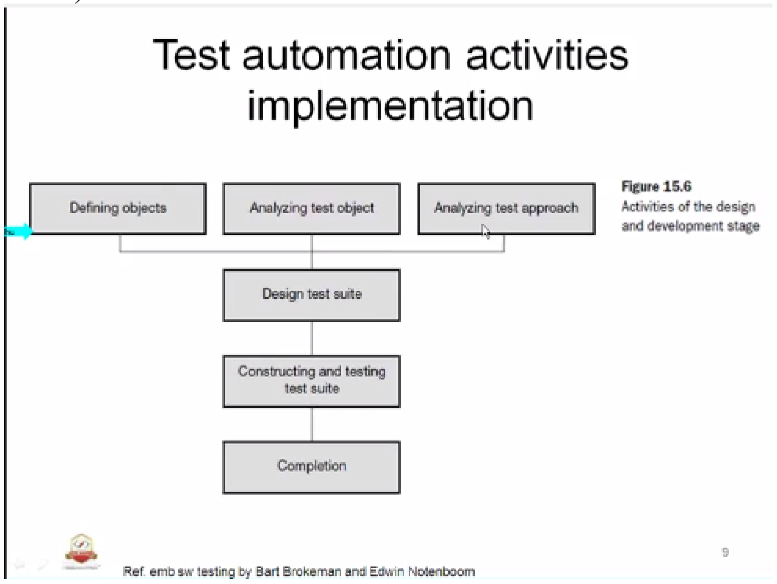
Figure 15.2 The technical implementation of the test actions is stored in a framework hidden from the user of the test suite

Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

7

Framework and the automation is being combined is this diagram which is also called as the blueprint of the test suit.

(Refer Slide Time 31:56)



Test automation activities implementation

Figure 15.6 Activities of the design and development stage

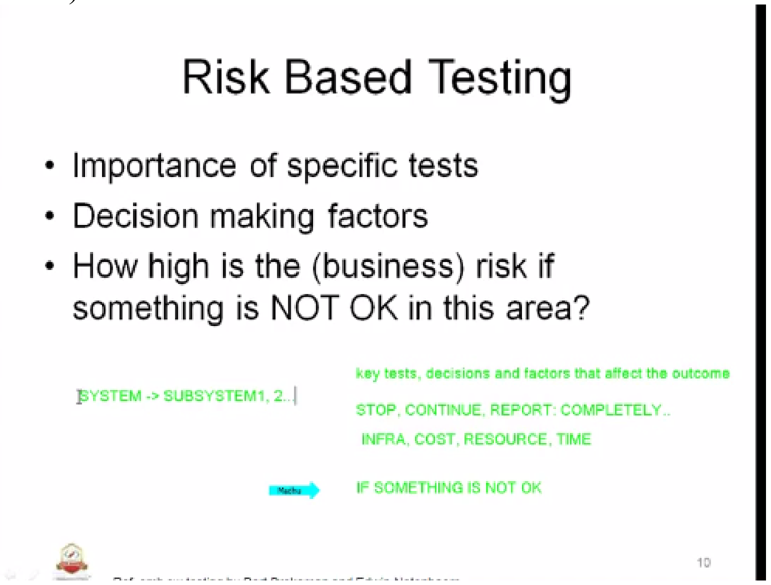
Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

9

In the next slide we see different actions or their activities how they are getting implemented in simple blocks as the design of the test suit are the adoration as use many blocks so you need define the objects and we need to analyses test objects and analyzing the test objects this is the first fundamental thing that we do then, with the help of this defined objects and analyzing the test objects and approach once you are finalized for each of the requirements for the particular suit then we are going to start the test suit we will start with the design of the test suit then, we will have the construction of the test suit with the scripts or whatever it is all that is done we reach the completion stage.

So the major object is to prepare and automated test ignorment capable of meeting the test automation objectives in order to realize of this some activities are defined, first is the

preparation of plan of action the purpose of being to show the activities, the people involved, time estimation etc… all is part of the first set of blocks.
The definition of the test automation object is detailed analysis of test object and test approach are very much necessary for the design decisions the decisions are also present here, what are the approach or any modifications are required for planning all this will be part of this. Parallel activities can be executed as necessary that could be similar functionalities where we do need a separate approach but straight modification in the inputs that also can be planned, so that it is very clear for the test or the test development thing to have what sought of approach for different blocks of the embedded system all these will be part of the implementation. So, that is about the test automation technic.
(Refer Slide Time 34:42)



Risk Based Testing

- Importance of specific tests
- Decision making factors
- How high is the (business) risk if something is NOT OK in this area?

SYSTEM -> SUBSYSTEM1, 2..

key tests, decisions and factors that affect the outcome

STOP, CONTINUE, REPORT: COMPLETELY..

INFRA, COST, RESOURCE, TIME

IF SOMETHING IS NOT OK

10

The next one is we will study about risk based testing, so what is risk based testing and why it is needed and some of the embedded systems used as an important in testing. So, basically so what we do in risk based testing is we will highlight or identify important of certain tests which are very keen in embedded software testing. Key tests and decisions and any factors that are going to affect the outcome of the entire embedded software testing, so that the decision can be further taken to stop the test or continue the test or report it or if it is a major revoke of the embedded system so that it has to be completely fixed, then risk based testing is very useful so basically, developing a risk based test strategy is communication that should be communicating to stack holders what is most important about this is for the product company or the organization are used for testing of this system. Instead of telling we do testing everything we will identify some of the key aspects of the testing and that could highlight some the business objectives also for the system under test, all this will be highlighted in this risk based testing for that there will be a strategy for those aspects of the test because there are lot of factors that are involved in terms of resources, resources could be people, money, time and infrastructure so, that also we will highlight and infra, cost human resource and schedule etc… so, if there is a major thing that needs to be taken care for this specific test all this will be strategized so, those are strategized under risk based testing the separate strategy that they will adopt for complex systems where certain functionalities have to be minimally working as a business objective.

So basically, in the structure test approach this is very important and it contributes to a more manageable test process qualities must be set and decisions made about what is important and what is not this all being specified.so important means what? It is very subjective concept of course with a risk based test strategy evaluating the importance is about answering the questions something like that.

How high is the business risk if something is not okay, in this test that means if something does not work while doing the testing so, what is the impact of that what is the importance of being that as not okay, so all answers for that will be part of the risk based testing and that should be communicated so, you will have define it so basically a risk is defined as the chance of failure occurred or occurring related to the damage expected when it is occurred so, what is the damage that is going to be the cause for that particular failure so, if the system is of sufficient quality which may imply high damage to the product or incense it could cause loss of market share.

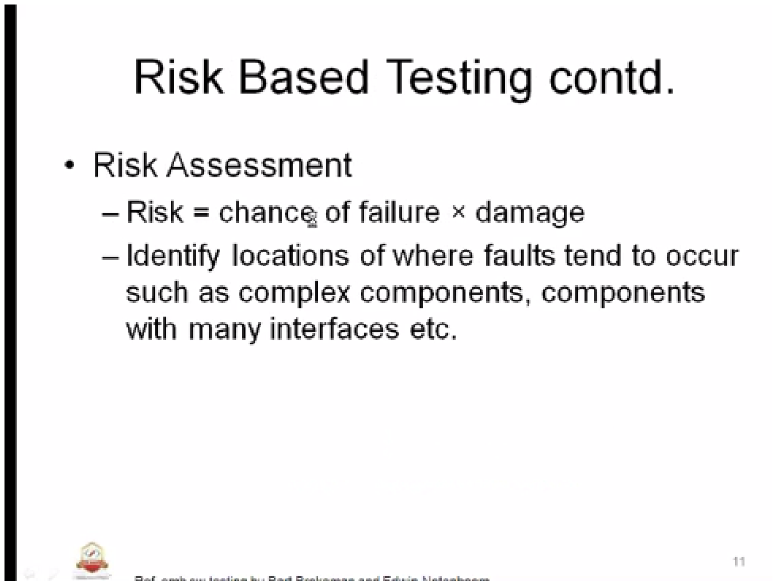
The company can be obelized to call back millions of sold products you have seen a call back of Toyota cars for a small issue that found near the break to replace that they have call that is called as call back millions of cars so it is going to be huge cost so those will be identified by while doing the strategy of certain aspects so in that aspects the importance of those key elements will be identified that is called testing that will be under tests, therefore the situation forms a risk for the organization whatever the highlighted elements which are called as risks.

So, testing of those aspects will cover such risks by providing inside into the extent which the system needs the quality demands. So if certain areas or aspects of the system imply high risks for the product and more thorough testing is obviously the solution so thorough testing of that particular aspects are important cause of the system is to be done those will be done from the quality stand point. Of course the other ways also is true where there is no risky is involved we may have to have a small bit of test or there is not test is required we do not have to do the test when there is no risk at all.

So how do we evaluate that there is no risk is involved or risk is involved if it is important or not important so this risk based testing strategy will answer that. The attributes would be from the quality or the performance or the product outcome in terms of business objectives. So usability and all that will be part of this risk based testing strategy and the sub systems also could be come under risk based strategy we will have a big system having subsystem 1, 2 etc… all this will be considered or failure attributes and risks and the importance of that particular failures and how they do is decomposition of the architecture the high level information system with the help of that they will arrive at the strategy decisions.

This specifically done with a help of quality attributes such as and do also they applied so is very important to have decisions making factors for such important aspects of the system so that is what is called as risk based testing.

(Refer Slide Time 43:11)



Risk Based Testing contd.

- Risk Assessment
  - Risk = chance of failure × damage
  - Identify locations of where faults tend to occur such as complex components, components with many interfaces etc.

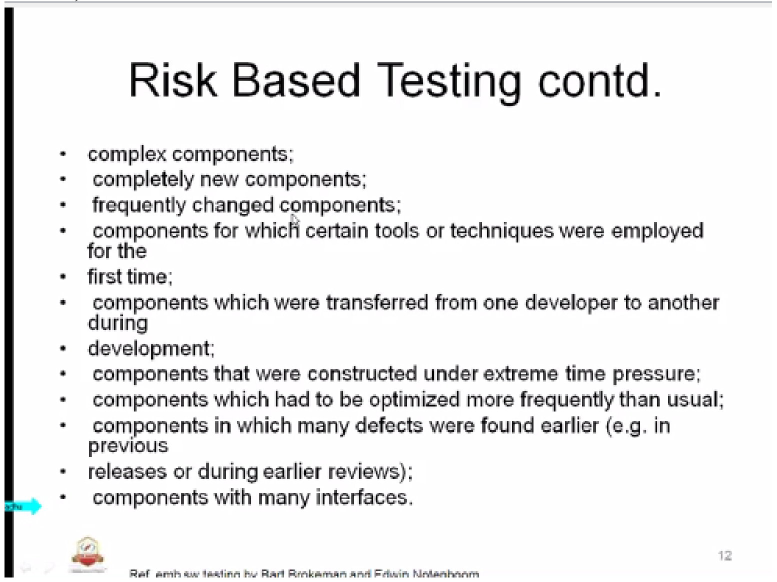
11

Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

Risk assessment has to be done for the identified risks so in chance of failure into the damage it is going to have is what is called as risk and that will identify the particular test strategy for type of risks that are identified so identify locations of where the faults tend to occur such as complex components, components with many interfaces signals etc…

Those need to be identified first and identification of what are the chances of failure and what is the cost of the damage so that will be arrived with the help of that risk will be accessed which is called as accessed which is very important part of the risk based testing.

(Refer Slide Time 44:05)



Risk Based Testing contd.

- complex components;
- completely new components;
- frequently changed components;
- components for which certain tools or techniques were employed for the
- first time;
- components which were transferred from one developer to another during
- development;
- components that were constructed under extreme time pressure;
- components which had to be optimized more frequently than usual;
- components in which many defects were found earlier (e.g. in previous
- releases or during earlier reviews);
- components with many interfaces.

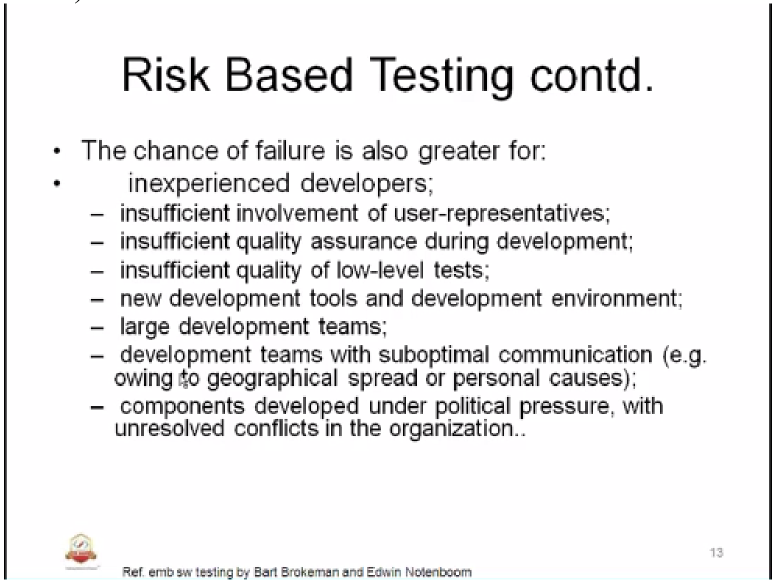
12

Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

And identification could be on some of these things like complex components completely new components the people are not knowledgeable or it is a new subsystem that has come to market that is been used that is a risk and if the components is getting changed in the embedded system could be embedded software could be a risk, that is also a risk components for which certain tools for techniques for who are employed for the first time and components which were transferred from one develop to another developer during the development with the chance that new developer would have injected from failures components that were constructed under extreme time pressure so very important term because the embedded software development this is the chance that last minute work will be more and due to delay some inputs the developer or detester due to high pressure, time pressure and time to market, the chance are for injecting the failures are more, and those components are need to be having the part of the risk components or risk elements, and here components means it is not the hardware components it is any piece of the embedded software system so components which had to be optimized more frequently than usual that means more optimization is required on those components that also can come under a risk.

Components in which many defects were earlier example previous releases or during earlier reviews we saw that certain components are having high defects still that has a high risk components with many interfaces being used in that components that is also a risk element.

(Refer Slide Time 46:09)



Risk Based Testing contd.

- The chance of failure is also greater for:
- inexperienced developers;
  - insufficient involvement of user-representatives;
  - insufficient quality assurance during development;
  - insufficient quality of low-level tests;
  - new development tools and development environment;
  - large development teams;
  - development teams with suboptimal communication (e.g. owing to geographical spread or personal causes);
  - components developed under political pressure, with unresolved conflicts in the organization..

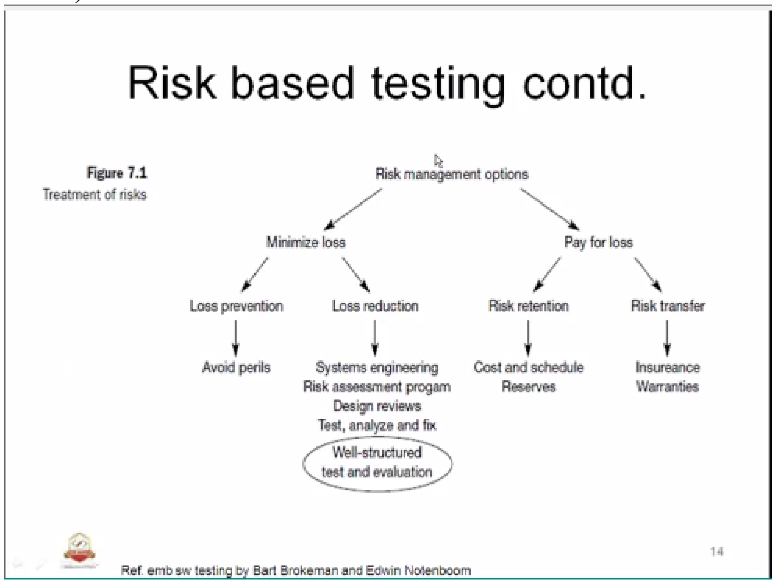
Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

13

So the chance of failure is also greater for the inexperienced developers insufficient involvement of user representatives; insufficient quality assurance during development; insufficient quality of low-level tests; new development tools and new development environment is also a risk large development teams were segregation is very difficult and interaction of different tools also is a risk developments with a sub optional sub optimal communication.

Example owing to geographical spread that means you have seen embedded industries spread across geographically different locations such as in aerospace industry we develop the entire system example bowing which is done in aortal USA whereas development is done in different location in USA and testing is done in India and fabrication and assembling is done in China so the development team or the testing team is completely disintegrated and interaction have to be sufficiently so as long as that is there the chances of failures are more due to which there could be slack in some of the communication or convincing the facts so there will be some hidden issues or errors that system can have components developed under political pressure, with

unresolved conflicts in the organization or shuffling the team also could be a strategy of risk basically.
(Refer Slide Time 47:59)



Risk based testing contd.

Figure 7.1
Treatment of risks

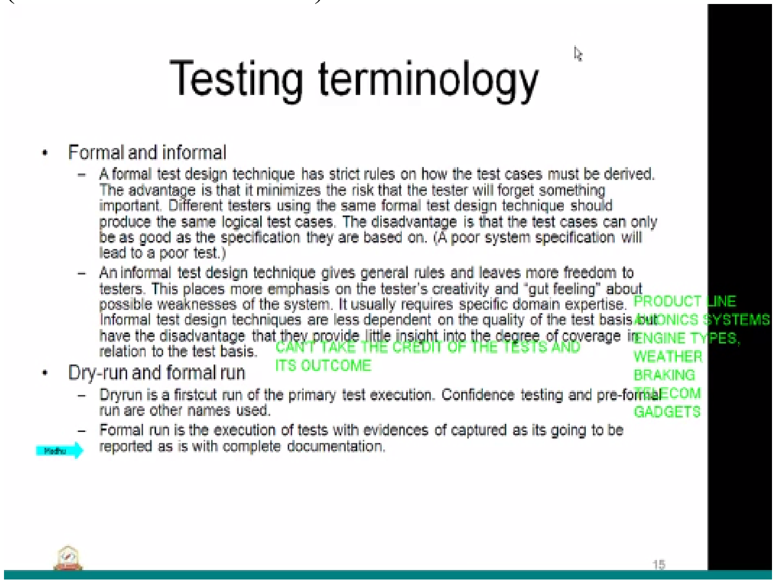
Ref. emb sw testing by Bart Brokeman and Edwin Notenboom

14

So that is about the risk based testing we come to these slide were how we can treat the risks that is where I don't call the risk mitigation as thing which is all management aspects will not study in detail but at least for understanding sake we will try to work on risk mitigation and risk contingency which we nauseate the risk and we will work out what best we can do in the contingency and disputation is which where it occurs what will be our approach.
So it is part of the risk management so what are the options that are available so categorization is something if this, so the required information for risk assessment is usually obtained from different sources and with the help of those sources we will arrive at different risk management options so, we have a broader categorization have minimize loss or pay for loss see here again the complexity which matters based on that the risk options are derived, so minimize loss is what one category of treatment of the risk there loss prevention and loss reduction and any threats we have will be done with the help of certain mitigation plan.
So that will work out for the loss prevention and in loss reduction system engineering risk assessment program design reviews, test, analyses and fix well-structured test and evaluation so all this will be part of the loss reduction so we are going to have a loss or certain failures all we reduce it, so these are all the aspects which can be treated for reducing the risk and as we progress toss embedded software testing we should try to reduce the risks that means loss prevention and try to avoid threats that are there for the particular system these are the one part were we are trying to minimize the loss and already the loss has occurred and the failures are there risk is realized then we have to pay for the loss.
So how are we going to de risk it or treat it so risk retention cost and schedule reserves that means we have reservation cost for that we are going to retain that risk and we are going to pay for it or we are going to transfer that risk so that somebody else can continue on this, it is basically a management thing and it is going to be communing or warranty and all that as part of that it is taken.

So risks are very high element important in embedded software testing as a management aspects, that need to be mitigated and that is about the testing and risk cases are testing aspects and strategy and we have a couple of important things that will be in embedded software testing method,

(Refer Slide Time 52:23)



Testing terminology

- Formal and informal
  - A formal test design technique has strict rules on how the test cases must be derived. The advantage is that it minimizes the risk that the tester will forget something important. Different testers using the same formal test design technique should produce the same logical test cases. The disadvantage is that the test cases can only be as good as the specification they are based on. (A poor system specification will lead to a poor test.)
  - An informal test design technique gives general rules and leaves more freedom to testers. This places more emphasis on the tester's creativity and "gut feeling" about possible weaknesses of the system. It usually requires specific domain expertise. Informal test design techniques are less dependent on the quality of the test basis but have the disadvantage that they provide little insight into the degree of coverage in relation to the test basis.
- Dry-run and formal run
  - Dryrun is a firstcut run of the primary test execution. Confidence testing and pre-formal run are other names used.
  - Formal run is the execution of tests with evidences of captured as its going to be reported as is with complete documentation.

15

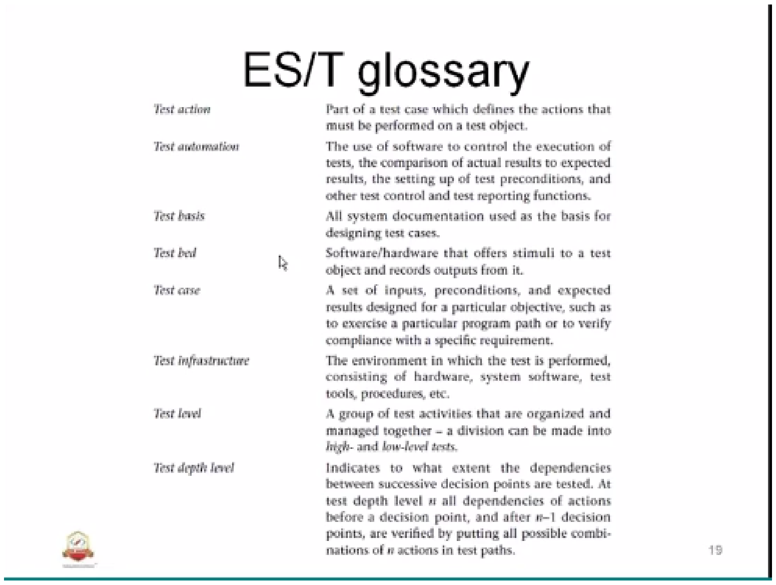
That is called the formal and informal testing so what is the formal and informal? In another aspect is dry-run and formal run so this two are used again and again in embedded software industry is that to understand what is formal and what is informal if formal test is in technique has strict rules on how test cases must be derived.

Documented or process oriented rules that will be documented reported and prozone, so strictly we are going to follow that in formal test design technique and execution the advantage is that it minimizes the risk that the tester will forget something important and all that because everything is based on grade out process.

Should different testes using different format design techniques should produce the same logical test that means we have a chain or a cycle that is well define the user will that and he do not want bother anything outside that so testes will use the same formal method or the deign technique again and again for doing the test and producing the results the disadvantage is that the test cases can only be as good as the specifications they are based on a poor system specifications will lead to a poor test because they have already committed doing the test and we do not have any second chance for the specification of the test.

So failures if it is going to happen, it is going to happen an informal test design is another one is a design technique that gives the general rules and leaves more freedom to testers that means informal test the tester has second chance and he can straight modify or there is a scope for improvement there are hook where he can adopt some changes are propose some changes there are what is that called templates where he can provide a proposal to the modifying it and still there is a chance that he can intro the product before the qualities is out for delivery so this places more emphasis on the tester creativity and gut feeling about possible weakness of the system that means the tester will have a full confidence of the embedded system it is usually

requires specific domain expertise as said it is very much it is important to have s system knowledge or domain, here domain means that is not just about the product and the product line where it is being used a product could be used in substances or systems it could be used in engine types to be used in whether forecast could be used in breaking because breaking into many, many some small substances so that are test what we are doing could be adopted that underneath that and we have telecom we have gadgets such as home entertainer or set of box whatever it could be so, all this will be requiring to have a thorough knowledge of the domain of that particular embedded series where the product is being developed tested and deployed.
So informal test design techniques are less dependent on the quality of the test basis but have the disadvantage that they provide little insight into the degree of coverage in relation to the test basis, that means everything will be informal the coverage we cannot take the credit test and its outcome simply because so whatever the tester has done in informal testing these all intermediate ones within of the prozone ones so they are the final ones, so that is where it have a disadvantage but, it is devised to have an informal test before we start the formal test that flow should be something like we begin with informal test and interfere all the get a good components on the entire system, understanding the domain apply the techniques inputted techniques optimization and anything that we want to do before we actually do the formal test.
So, they use it in the embedded system industry formal and informal that are used, you see coolant of that are being used in specific industries that is called dry-run and formal run the informal run is called as a dry run that is at the first cut run of the primary test execution so basically, it gives a confidence and any re visit is required time turning is required all will be done with the help of this dry run which something like an informal or pre formal run and there are different means that are used but, generally in the embedded industry they use it as trizonement and formal zone we know it is same as whatever studied in the previous visit the formal run is the execution of this with evidences captured as it is going to be reported with the complete documentations.
So, once we are done with this formal run that is what is going matter on terms of reporting pass fail revisit or whatever it is required so, that is about the testing terminology used in the embedded industry, called formal and informal and trizonment informal run. So with that we come to end of unit one so probably we will touch base what are the chapters we visited so far in the next session so before concluding the class I will go through some of the glossary,
(Refer Slide Time 59:37)



ES/T glossary

Test action	Part of a test case which defines the actions that must be performed on a test object.
Test automation	The use of software to control the execution of tests, the comparison of actual results to expected results, the setting up of test preconditions, and other test control and test reporting functions.
Test basis	All system documentation used as the basis for designing test cases.
Test bed	Software/hardware that offers stimuli to a test object and records outputs from it.
Test case	A set of inputs, preconditions, and expected results designed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.
Test infrastructure	The environment in which the test is performed, consisting of hardware, system software, test tools, procedures, etc.
Test level	A group of test activities that are organized and managed together – a division can be made into *high-* and *low-level tests*.
Test depth level	Indicates to what extent the dependencies between successive decision points are tested. At test depth level *n* all dependencies of actions before a decision point, and after *n*–1 decision points, are verified by putting all possible combinations of *n* actions in test paths.

19

that we have for today, test action part of a test case which defines the actions that must be performed on a test object, test automation we have seen the use of software to control the execution of tests, the comparison of actual results to expected results, setting up of test preconditions ,and other test control and test reporting functions. Test basis all system documentation used as the basis for designing test cases. Test bed software hardware that offers stimuli to a test object and records outputs from it.

Test case we know that it is set of inputs, preconditions, and expected results designed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement you know that requirements level we have a high level test for the black box testing and we have test cases accordingly and for white box testing we will do the coverage and all that while do the execution on the implementation or the code.

Test infrastructure the environment in which the test is performed, consisting of hardware, system software, test tools, procedures, etc… with the help of infrastructure we develop or build the environment also have the automation aspects in that. Test level we have seen in previous sessions a group of test activities that ate organized and managed together-a division can be made into high-and low-level tests.

Then we have test depth level indicates to what extent the dependencies between successive decision points are tested. At test depth level at all dependencies of actions before a decision point, and after n-1 decision points, are verified by putting all possible combination of n actions in test paths. So that is about the glossary that we saw, so with this we come to the end of this class in today's session and the units that we have completed unit one and unit two we will give a small recap of whatever the units and sessions we have completed and we will take up the next unit that is static analysis, inspection and reviews of the next session.