

Welcome you to the last session of unit 5 embedded software testing and this is the last theoretical session of embedded software testing and in this unit we will try to study the test management aspects in terms of defect management test management and control and test management tool

(Refer Slide Time: 00:25)

Test Management & control

What to do when things happen that affects the test plan:

- Re-allocation of resources
- Changes to the test schedule
- Changes to the test environment
- Changes to the entry/exit criteria



1

Defect management tool.

(Refer Slide Time: 00:27)

Test Management & control

Test Control

What to do when things happen that affects the test plan:

- Re-allocation of resources
- Changes to the test schedule
- Changes to the test environment
- Changes to the entry/exit criteria
- Changes to the number of test iterations
- Changes to the test suite
- Changes of the release date



2

Then after this

(Refer Slide Time: 00:27)

Test Management Tool

- The test management tools provided by tool vendors offer only the functionality to store test cases, scripts, and scenarios and sometimes integrate defect management.
- They have no facilities to store test plans and project schedules. This type of test management tool is not very useful for planning and control activities.
- Tools with the ability to link system requirements to test cases are now available.
- These tools afford the ability to keep track of the coverage of test cases in relation to the system requirements. In addition, they become very useful if system requirements are changed or might change – the test manager is able to show the impact of these changes and this may give cause to block the changes



2

(Refer Slide Time: 00:28)

Defect Management tool

- 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, for instance, the status of all the defects or the ratio of solved to raised defects. Several tool vendors have developed defect management systems.
- All these are designed around a database system and have tracing and reporting facilities.
- The tools have the ability to implement a workflow with security and access levels. Most have e-mail facilities and some of them are web-enabled.
- A defect management system is used to store defects, trace them, and generate progress and status reports. The tracing and reporting facilities are almost indispensable as control instruments



3

We will go through all the units what we have to do quickly. So, that we are done with all of the typical parts of embedded software testing. So, before this we will just go through what we have understood in the previous session about test management. So, we had gone through (Refer Slide Time: 00:56)

Change Management

- Document the change request
 - Analysis of impact
 - Approval of change request
 - Major changes by CCB
 - Minor changes by Project Manager
 - Making of changes
 - Check-out of configuration items from baseline repositories
 - Reviewing as required
 - Check-in of changed configuration item with new version number
 - Change Closure (through PM updates)



1

Change management. Document the change request, analyze the impact, approval is required from CCB change control board, who will approve for the changes after the analysis plan and final approval with the senior management plan and many minor changes will be taken for within the – without highlighting to the higher management. So, making of changes will be gone through check-out check-in process.

Which is nothing but the configuration control process, were we check out, we will take out from the repository and once we update all the reviews are done? We are going to check in back so that the new version is created. And the changes are available in the repositories. So that is how we do the changes onto the corresponding document code, data aspects, test cases anything that are part of the test cases.

Sorry they are the part of the configuration items CS. Then we are going to close that change particular change saying that it is based lined or it is available for the release or delivery. That is what the change closure.

(Refer Slide Time: 02:11)

Incident Management

What is an Incident?

- Any significant, unplanned event that occurs during testing or other event that requires subsequent investigation or resolution.
 - Differences in actual and expected test results
 - Possible causes:
 - Software fault
 - Expected results incorrect
 - Test was not performed correctly
 - Can be raised against code and/or documents

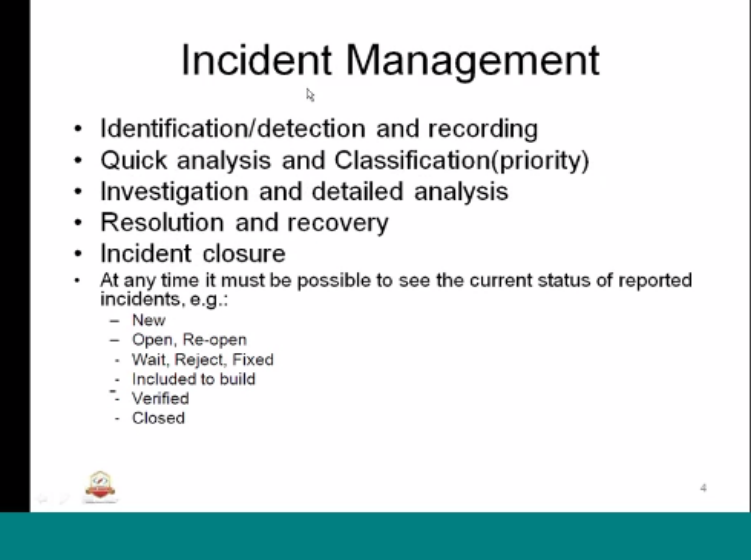
12



3

Then we had gone through incident management. So you have understood what is an incident. Any significant unplanned I went that is of course during the testing or any other event that requires some investigation under solution of that particular incident. So, the incident could be due to a fault intermediately it has happened unexpected. So, the results are incorrect. Test was not performed correctly.

So, we can raise the issue incident against the unexpected behavior or it could be self whatever that particular tests are we may trying to test it as well because testing may be a incorrect way of doing of that. That is also can be called as the incident. So, that there is a need for correction of the incident or the resolution for that particular incident. So, in incident management (Refer Slide Time: 03:07)



Incident Management

- Identification/detection and recording
- Quick analysis and Classification(priority)
- Investigation and detailed analysis
- Resolution and recovery
- Incident closure
- At any time it must be possible to see the current status of reported incidents, e.g.:
 - New
 - Open, Re-open
 - Wait, Reject, Fixed
 - Included to build
 - Verified
 - Closed

We first identify the defect and record what is the incident about and details of that entire incident. Then we do an analysis and classification priority like whether this five incidents have happened? How we can prioritize? What is the impact in that? Whether instruction stops for the rest of the program to continue? Or to test execute of all these? So, next one is the investigation and detailed analysis.

We will try to investigate what is the cause? Where the problem was? Before that incident cause incident had happened. And we try to put a mechanism for recovery how we can recover that particular incident? Whether we can fix that code? Fix that design or requirements or the test approach itself is wrong. All this will be in the part of the resolution and recovery. Then we will have a different stage of incident like re-open, wait, rejected, included to build, verified, closed etc. and every revision

(Refer Slide Time: 04:11)

Revision History

Rev. No.	Date	Author	Approval	Changes
1.0	23/May/2012	Madhu	Arvind	1. Updates as per new template 2. Updated section 2.3.2 for the scenario changes 3. Appendix added...
2.0				



5

That we are going to release for the deliver as an – should identify the clear history of what is going on in the particular CI cor-configuration in term. So, every configuration item is suppose to have a something like this mandatory which will identify the particular revision number why it is what revision number it is been numbered and on which date it is been revised, who is the author?

Who is the reviewer or the approver? And what are the changes that particular revision number has undergone. Similarly up on the next change we are going to highlight it as the revision number 2, 3, 4 like this. Each one will identify clearly. So, what is the revised detail or the updated details or the chain details that are undertaken within the particular region? That is what we do.

That is about the revision history how we are going to revision for each of the CI.
(Refer Slide Time: 05:08)

CM Tools



- All testware has to be stored in a proper way. After quality checks, the testware should be frozen.
- The next step is to make the testware configuration items and store them in a configuration management tool.
- In addition to the regular testware (test cases and test documentation) the description of the test environment, tools used, calibration reports, and specially developed stubs, drivers and simulators should be subject to configuration management.
- A configuration management tool provides the functionality to keep track of the configuration items and their changes.



6

Then for understanding the configuration incident management and all that, we went through the need of CM tools. So, all the test were has to be having identified in a physical location or a

particular repository. So, to control and manage that we need CM tool or a configuration management tool such as the, incase (Refer Slide Time: 05:34)

CM Tools

- PVCS Dimensions
- MKS
- SVN
- ..

PVCS dimensions, SVN etc. so this basically used for the CI. It could be a development it could be a testing could be a test case design, any documents, any access anything that is the part of the project. All is will be controlled and managed through configuration management tool. Against of the configuration management tools we try to understand and (Refer Slide Time: 06:02)

CM Tools PVCS Dimensions

Content Window

Specification	Description	Dimensions	Status	Update Date
PAYROLL_DEV_REL_3	Baseline DEV	1	UNIT TESTED	20-Jun-2005 17:19:52
PAYROLL_DEV_REL_2	Baseline DEV	1	UNIT TESTED	20-Jun-2005 17:19:50
PAYROLL_DEV_REL_1	Baseline DEV	1	UNIT TESTED	20-Jun-2005 17:19:48
PAYROLL_REL_1	Baseline REL	1	APPROVED	20-Jun-2005 17:19:48
PAYROLL_REL_2	Baseline REL	1	APPROVED	20-Jun-2005 17:19:48

It has various windows which will highlight different colors and terms of - . When it was updated within the particular project and whether it is in line, whether it is post lined and if it is base line how many elements of that particular baseline is ready for- etc. all this will be the part of the tool as it displays you can see it. (Refer Slide Time: 06:27)

Test Management

- Planning of the test project



9

You can configure the tool also it does not matter. Usually the tools will allow user to configure the way that project has to be fashioned. Then we understood about the test management in terms of planning of the test project.
(Refer Slide Time: 06:44)

Test Management

Activity	Start	End	TM	MS	TS	DE	TCM	TST
Test plan	07 08	11 08	32					
Planning and control	14 08	04 10						
Test management			48					
Test configuration management						28		
Methodological support				44				
Preparation phase	14 08	15 08						32
Specification phase	16 08	08 09						186
Execution phase	10 09	26 09			30	50		170
Completion phase	27 09	28 09						16

Table E.5

Planning of acceptance test "James" and "Mother"
TM = test management, MS = methodological support, TS = technical support, DE = domain expert, TCM = test configuration manager, TST = tester



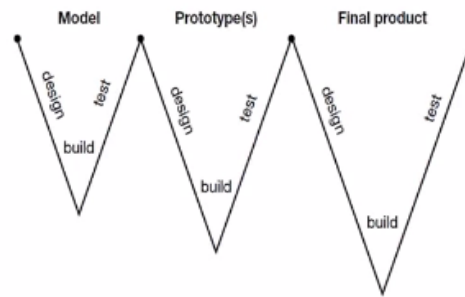
10

Various aspects of test management and its effort in the test management methodological support, technical support, domain expert, test configuration management and tested all these efforts for each of the activity and the test management will be updated. It is start and ended something like a planning document or a management document in this tracking will be done, how much of effort is gone here and all that.

And in one of our session we saw about the variance like, this is the plan this is the management plan against which how much we have consumed? How much is left over? What is our trend? And what is the variance for how much schedule? What is the variance on effort? All this will be the part of the test management. We are going to take care. Now we
(Refer Slide Time: 07:37)

how the test process relates to the software V-model

Figure 3.1
Multiple V development lifecycle

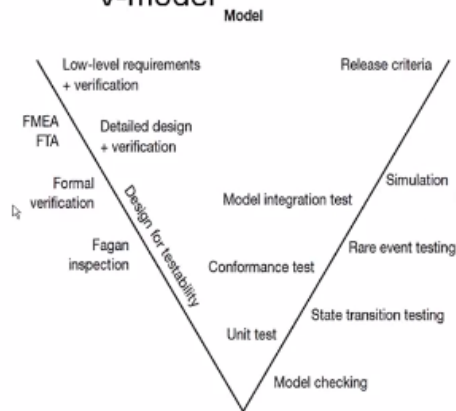


11

So, we studied about how the test processors are related to the software v-model. We understood about the multiple v development life cycles. You can see model prototype as well as product and each as its own v model and that is why it s called as multiple v development or life cycle model. And that life cycle model as on the right hand side edge as a test aspects or test activity. And that is on for with the left hand edge of the particular model or prototype or any product development. That is how it is aligned with the v model
(Refer slide Time: 08:16)

how the test process relates to the software V-model

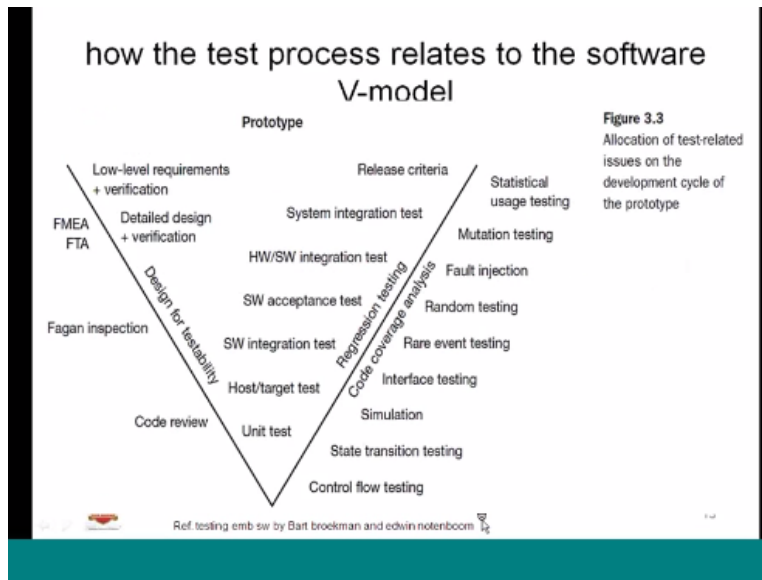
Figure 3.2
Allocation of test-related issues on the development cycle of the model



Ref. testing emb sw by Bart broekman and edwin noterboom

12

In detail the same thing is done explained with the program. It is from that book called testing embedded software by Bart Brookman and Edwin noderboom.
(Refer Slide Time: 08:30)



Models are and we are gone through this in our one of our earlier sessions. I think it is in unit 3. So, similarly the different v-models various v-models we have gone through. And the right hand side you can see how the testing is aligned in the rare event testing, random testing, statistical testing, certification all that. Ultimate aim is to read the right most, right edge of the upper side of the v-model.

Basically so release that particular activity which is unified by other particular v-model and we are going to define release criteria. The next type of testing (Refer Slide Time: 09:17)

Testing – Design by contract

- An approach that uses the documentation only to capture the design but also to encourage interaction among developers.
- In design by contract, each module has an interface specification that precisely describes what the module is supposed to do
 - Meyer (1997) suggests that design by contract helps ensure that modules interoperate correctly
 - This specification, called a **contract**, governs how the module is to interact with other modules and systems
 - Such specification cannot guarantee a module's correctness, but it forms a clear and consistent basis for testing and verification
 - The contract covers mutual obligations (the reconditions), benefits (the postconditions), and consistency constraints (called **invariants**)
 - Together, these contract properties are called **assertions**
 - Testers care about how this contract is enforced

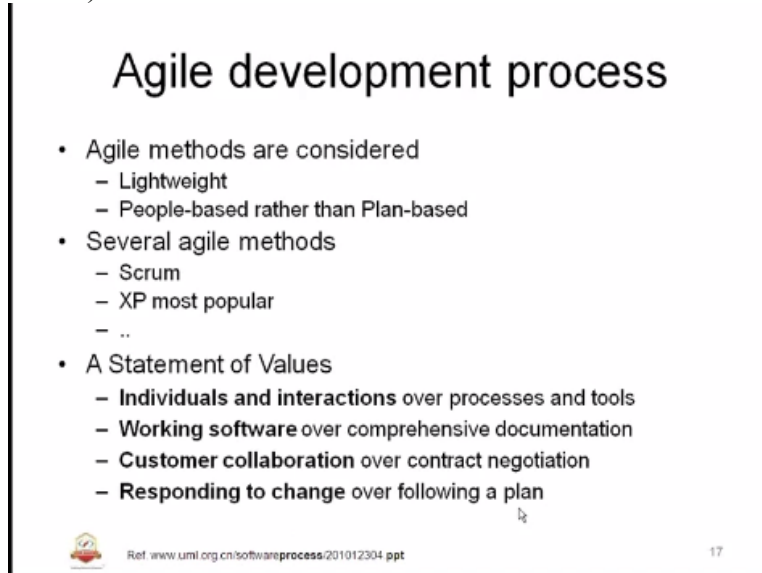


For as per of the interest in the software testing is testing design by contract. Here the approach uses the documentation only to capture the design but it will encourage basically sort of the instruction among different developers. So, that particular developer is allotted work like this contract.

And they will provide the support in terms of interacting each other and providing the solution. So that is so we do the test. This suggestion called as the interact as per contract and that contract

will be developed after the instruction is done with the different module owners and the system owners.

(Refer Slide Time: 10:06)



Agile development process

- Agile methods are considered
 - Lightweight
 - People-based rather than Plan-based
- Several agile methods
 - Scrum
 - XP most popular
 - ..
- A Statement of Values
 - **Individuals and interactions** over processes and tools
 - **Working software** over comprehensive documentation
 - **Customer collaboration** over contract negotiation
 - **Responding to change** over following a plan

Ref: www.uml.org.cn/softwareprocess/201012304.ppt

17

Then we had done through a process called agile development process. They basically use in short duration time to sort of a application, embedded application embedded prisms, which is totally away from normal and allotting process so the current processor are two heavy weight to comparison in the documentation process and all that, so all that will be by passed agile development process.

And the current software development is to resume is difficult items of changing requirements, and in competition, short development cycles all is very difficult to align with the current model, so the process itself are deviated in the terms of doing agile, where it is totally align with the business process or a value to the business basically it brings up, and different methods agila, such as extreme programming, scrum like this.

So, the most popular one are the member XP, the simpler programming, so the basically values are assembling like individual interaction over process in tools, the working software was comprehensive documentation, customer collaboration we take more in the agile process, responding to change over following plan, so as soon as the chain comes we going to start working in the implementing all that. Instead of going to the plan process analyzing all that, there is no time for all us so, that process mechanism.

(Refer Slide Time: 12:03)

Agile development process

- Agile methods:
 - Scrum
 - Extreme Programming
 - Adaptive Software Development (ASD)
 - Dynamic System Development Method (DSDM)
 - ...
- Agile Alliance (www.agilealliance.org)
 - A non-profit organization promotes agile development



Ref. www.uml.org.cn/softwareprocess/201012304.ppt

18

Extreme programming adapt you a software development and dynamic system development method all are the types that we have, you can see more details non-profit organization they define and they promote agile development basically companies like NHP or any computer electronics group, they follow this tool to develop the product types all that, they adaptive agile scrum more development process, for testing also part of the scrum development.

(Refer Slide Time: 12:33)

Agile Development Scrum process

Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.

It allows us to rapidly and repeatedly inspect actual working software (every two weeks to one month).

The business sets the priorities. Our teams self-manage to determine the best way to deliver the highest priority features.

Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance for another iteration.



Ref. www.uml.org.cn/softwareprocess/201012304.ppt

19

So they will identify any mechanism to align with the agile development.

(Refer Slide Time: 12:41)

Agile and Scrum Process



Ref. www.uml.org.cn/softwareprocess/201012304.ppt

19

So, you can see that example what I put there, and they call it as string back lock, so every day they will base on the frequency, that the back locks are cleared and identified and it is taken here, so for example it is a 30 day project, within 24 hours there is a action for ridge of the development or the stakeholder, they will act on that, there is the dead line.

(Refer Slide Time: 13:11)

Agile Testing Scrum process

- What are roles of tester in Scrum? There is NO tester in formal Scrum Process
- Testing is carried out by developer with Unit Test
- Testing coverage : Testing is carried out by Product Owner/ Client
- Frequently testing by Product Owner – each sprint • Testing Acceptance Criteria



Ref. web

20

As I said the real testing from the process also will be used, there is no specific dedicated tested in the formal scrum process. Testing is carried out by the developers with the in testing mechanism; testing coverage is carried out by the product owner or the client, client retakes care of power and all that by seeing the report bigger as the specification.

So, the testing is taken care by each tool something like spring back lock or whatever it is as for the definitions. And there is a acceptance criteria for big effort. And that you will be taking care.

(Refer Slide Time: 13:50)

Test Driven Development

- **Write a Test That Fails**
- **Write Just Enough Code**
- **Repeat**

Test Driven Development means testing executed before development and refactoring performed to optimize development



22

Next type of thing is test driven and development. So basically we write the test for the specification and we will try to develop the project. So basically the project is oriented towards the test or the test driven, surrounding the test driven cases in the object at the development. So we write a test case that fails and write just enough code for that to fix the fail mechanism, the complete testing is taken here and the development is done.

(Refer Slide Time: 14:33)

Test Management & control

Test Control

What to do when things happen that affects the test plan:

- Re-allocation of resources
- Changes to the test schedule
- Changes to the test environment
- Changes to the entry/exit criteria
- Changes to the number of test iterations
- Changes to the test suite
- Changes of the release date

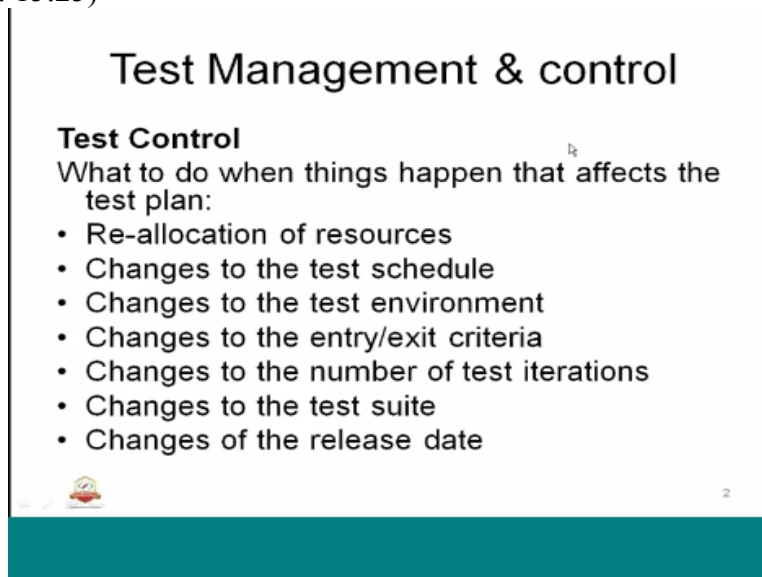


23

So test management and control, what to do when things happen that effect the test plan. So if is hit the test plan, we are going to definitely work on the test mechanism and we need to control it, so we need to re-allocate it for every resources, we need to change the schedule, we need to change the environment.

We need to redefine the entry exit criteria, and number of iterations changes we need to take care and test suit related changes are there, and so we need to see the effect and the relegate we need to postpone or update. That's what we have studied in the earlier session3; today we read and quickly understand the test management

(Refer Slide Time: 15:25)



Test Management & control

Test Control

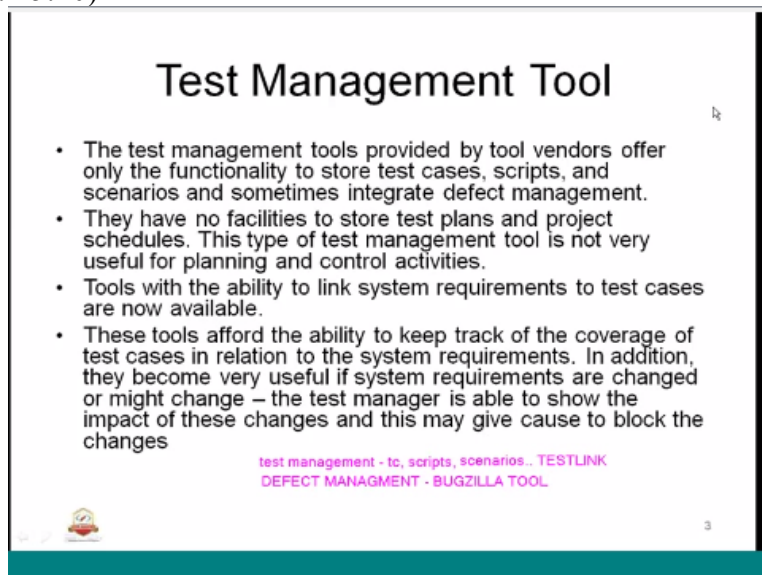
What to do when things happen that affects the test plan:

- Re-allocation of resources
- Changes to the test schedule
- Changes to the test environment
- Changes to the entry/exit criteria
- Changes to the number of test iterations
- Changes to the test suite
- Changes of the release date

2

And in the terms of a test management tools so we know that reallocation of the big sources for entire test cycle defects, it will manage and control.

(Refer Slide Time: 15:40)



Test Management Tool

- The test management tools provided by tool vendors offer only the functionality to store test cases, scripts, and scenarios and sometimes integrate defect management.
- They have no facilities to store test plans and project schedules. This type of test management tool is not very useful for planning and control activities.
- Tools with the ability to link system requirements to test cases are now available.
- These tools afford the ability to keep track of the coverage of test cases in relation to the system requirements. In addition, they become very useful if system requirements are changed or might change – the test manager is able to show the impact of these changes and this may give cause to block the changes

test management - tc, scripts, scenarios - TESTLINK
DEFECT MANAGEMENT - BUGZILLA TOOL

3

So, that is what we do with the test management tool. so the tool will basically take care of all the artifact and it will highlight and it will provide the tracking mechanism in terms of where we are testing that, so that report will be used and it will be tested and adopt for the strategy that going to work out for the a release, so basically the test management tool provided by tool vendor of only the functionality to store the STFF.

Scripts and scenarios and sometimes the integrate defect management, something like bugzilla, test management example I will try to tell you, what we been used in our one of the projects. Test management identifies the test cases, scripts, scenarios, all this will be there and taking with the test management in to such test links. Take one of the session practical sessions, we will go through this, how these tools work.

And we will try to create a big size sample projects, identifying the test management and test case in scenario. Similarly we need to identify defects using defect management tool, all these will be basically used for managing the test something like bugzilla tool is used for the test management and the defect management. So basically it offers to store all this artifacts, scripts, scenarios, bugs and all.

And they will not store plans, so plans test manager has to see and this tool will provide the numbers and all values like artifacts and we need to align that ASP to the projectors, so they have no facilities to store plans and projects, but mistake of test plan is not basically useful and controlling their activities, but there need to be aligned manually, the test manager that's what they do.

But the good thing is that tools will have the ability link system we call in the test cases, such as test links applicant link, and how many test cases have been failed, resources have been failed and passed etc, will be taken care. So these tools afford the ability to keep track of forage of preparator, how many test case have been passed in the coverage report again is the system requirement specification requirement.

These tools will provide. In addition they will become very useful or if it changes like test manager there will be surely in fact of this changes and how many prospectors we have to execute or they execute once requirement changes, so that, those things can be highlighted with the tool basically and report according. And the show toppers in the terms of very easier issues or the bugs that, which have been uncovered all this will be highlighted with the test management tool. The other part I said that it is
(Refer Slide Time: 19:47)

Defect Management tool

- 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, for instance, the status of all the defects or the ratio of solved to raised defects. Several tool vendors have developed defect management systems.
- All these are designed around a database system and have tracing and reporting facilities.
- The tools have the ability to implement a workflow with security and access levels. Most have e-mail facilities and some of them are web-enabled.
- A defect management system is used to store defects, trace them, and generate progress and status reports. The tracing and reporting facilities are almost indispensable as control instruments

The defect management tool such as bugzilla can be used; the defects detected during the test process must be collated in an orderly way basically, because all the defect need to used and control, that is why this tool is useful. For the small project a simple files can with few control processor efficient, one request the other or documentation is enough and we will highlight it manually.

But as we progress throughout the entire project, we cannot be sufficient to maintain through simple system mechanism tool, we have to leave with the complexity and forget which advice is to use the defect management tools. Tools such as bugzilla, it is really in industry they have tools defined actually, bugzilla is the royal to free , it is not a royal to free thing, so if commercially we want to use it we have to pay them more through the royalty license system and the corporate.

But in general what happens is, in the build software industry where they use their own in-house tool to maintain the defects and detect artifacts, and again is that management in to collect it and report to the customer and all, so the complex project need that, this data base should be possibility of stimulating the progress report showing for instance status for all groups or ratio of all race defects.

Several tool vendors have developed the defect, all these are designed around the data processing and having tracing of the management tools, basically all the depositary which we got defects to be stored in the data base system and that data base system will be smartly used by the UIS and there are useful report and can be used to track it, the tools have the ability to implement our security and accessible so the admin can define who can secure the files, who can put in to the depositor.

Who can updated that work flow or work instruction can defined. So that the individual having the different levels of accessible, developers can have a access for develop the various level and testers can have a development can have a development access, they cannot read, they cannot write, and all this sort of a control mechanism and they will be defined by the admin for using the defect management.

And also as long as a defect are opened closed and progress hold and we can setup in your typical position in the terms of email facilities and some of them are web enabled, so you can trigger that in the web itself so that once the stages have been moved automatically communication will be happening to the respective stakeholders. The defect manager system is used to store.

The defects traced in generate of this instruction, the addressing and reporting facilities are more in this principal as control instruments, so, very important aspect of the defect management. So with that we come to the conclusion of the test management tool and defect management tool, with this session, so with this we are going to end the embedded software testing large, now we will try to. Okay so this next few minutes I am going to discuss and highlight about what we have seen overall in the embedded software testing, right from the unit 1 to unit 5.

(Refer Slide Time: 24:10)

Unit 1: Introduction to EST and fundamentals of testing

- Embedded system basics
- Embedded system environment
- Embedded C glance
- V&V (Verification And Validation)
- Cost of embedded SW defects
- Test process basics
- Embedded software system testing basic setup



So, the units are divided such a way that all the aspects of embedded software testing which takes place and unit 1 we studied regarding which one? Fundamentals of EST embedded software testing. Right, that is the first session we had and in the second session we understood about testing methods third one, is on static analysis.

And the code reviews, so the fourth one is basically software integration, fifth one is nothing but test management because each these five sessions are very five units are very important part of the embedded software testing. So, that is how it is been because it is been divided that way. We will try to go through as pointers what are the things that we have identifiably in this. So we started with fundamentals, then different testing methods, dynamic white box black box and all that. Next one is on the offline part of the static analysis for reviews, types of reviews and all that. Then we understood about hardware software, software integration methods and regression all that. The last one is about the test management in terms of test planning, test life cycle alignment with the development life cycle and tools that are use for managing the tests as well as the defects.

So, that is how the inter course is being parted and we have gone through that. Okay, so in unit 1 it was about fundamentals of testing. It was in directory session to embedded software testing and we also gone through embedded system basics, embedded system environment, embedded C glance because embedded C code is the specific code.

And it is the separate course of course by itself and we try to understand the basics of that and compilation linking and all that basic parts of that embedded C language and the embedded system software. We understood in that class why I took embedded C, so the most of the 90% of projects that are there in the world for embedded system or on the eloper basically using the C language.

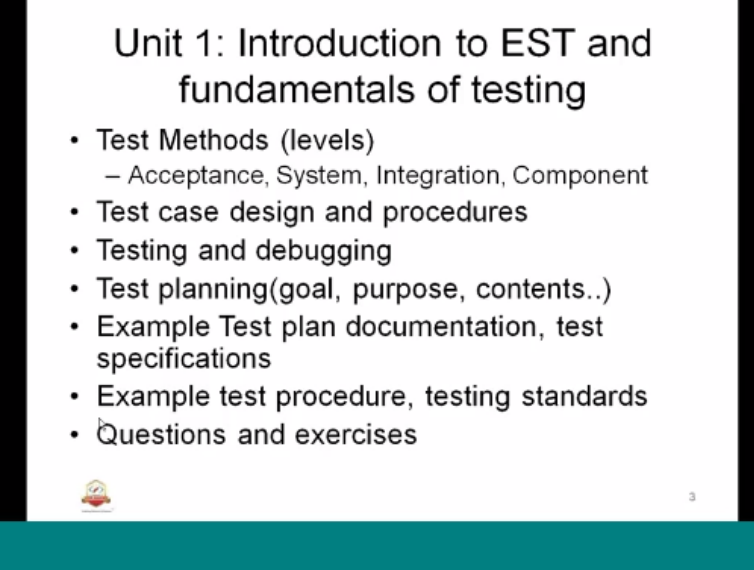
So, that is why it is very important to understand. Embedded system is basic with the C, embedded C language and fundamentals about that. Then we had studied about V and V validation and verification. Cost of embedded software defects. We have seen in the graph

plotting the various stages of software. Embedded systems we find defects it will be difficult to control as you progress for the end of the project.

So, it has to be fixed in the earlier stage. So, that is why we need to have a right sort of a testing mechanism. So, that is what we understood about the complexity. Then test process basics, what are the different processes that are followed like test cases, - and the scripts and the execution on the environment all this are basics. And embedded software that system testing basics set up how I going to have a setup.

So, we had a contra setup how the system is communicated with the target board as a white box how it is communicated with the target board or the black box and how the recruits are been used for the scripts that are driven. All these basics we understood with the good example of a diagram and we also understood about the develop environment testing environment as well.

(Refer Slide Time: 29:08)



Unit 1: Introduction to EST and fundamentals of testing

- Test Methods (levels)
 - Acceptance, System, Integration, Component
- Test case design and procedures
- Testing and debugging
- Test planning(goal, purpose, contents..)
- Example Test plan documentation, test specifications
- Example test procedure, testing standards
- Questions and exercises

3

So, continuation of the embedded software testing fundamentals tests methods. We have gone through the acceptance, system, integration and component level testing the various levels of testing. Then principles of test case is designed and procedures how it looks like with the example all we had gone through and how we can do that testing and debugging on the target and on the system we had gone through.

Test planning like what is the goal purpose, what are the contents of the test planning in the planning process. We had gone through then we studied about the example test plan documentation and test specifications also we try to know. Example test procedure also we had gone through testing standards in terms of guideline and rooms and all that. How the testing document should be developed and testing should be carried out. You understood? Also we had few questions and also exercises we have made in the first unit.

(Refer Slide Time: 30:17)

Unit 1: Introduction to EST and fundamentals of testing

- Depicting levels of testing
- Unit, integration, system and acceptance testing basics
- Definition of test harness, test bed, setup..
- Host and target based systems
 - During development stage
 - During testing stage
- Target based debugging and testing
 - Simulation, emulation, target monitoring
- EST tools categorization, listing as part of planning
- EST setup – Software environment configuration, tool snapshot
- Definitions of entry and exit criteria for EST



4

And we had drawn a block depicting various levels of testing and those levels are unit, integration system and acceptance testing basics, definition of test harness, test bed, setup. Host and target based systems. During development stage how it looks like during testing stage how it looks like then target based debugging and testing. We have a three various types of them simulation, emulation and target monitoring.

Each of them we try to understand and advantages and disadvantages of them and colaberately we are going to use it for the embedded software testing. And tools what are all the tools? Entirely the embedded software testing that are used or categorized and we try to put a list and a snapshot of embedded software testing tools and it will be listed as part of the planning, test planning.

And setup software environment configuration and tool snapshot we had gone through definitions of entry and exit criteria for embedded software testing, the various cases that goes. (Refer Slide Time: 31:32)

Unit 1: Introduction to EST and fundamentals of testing

- T-Emb method
- LITO principles
- SW Life cycle, entry/exit criteria
- Prototyping life cycle and example
- Formal life cycle and example
- V-Model life cycle
- Embedded Life cycle processes
- Life cycle process example of Consumer electronics
- Automotive embedded projects' testing phases and process

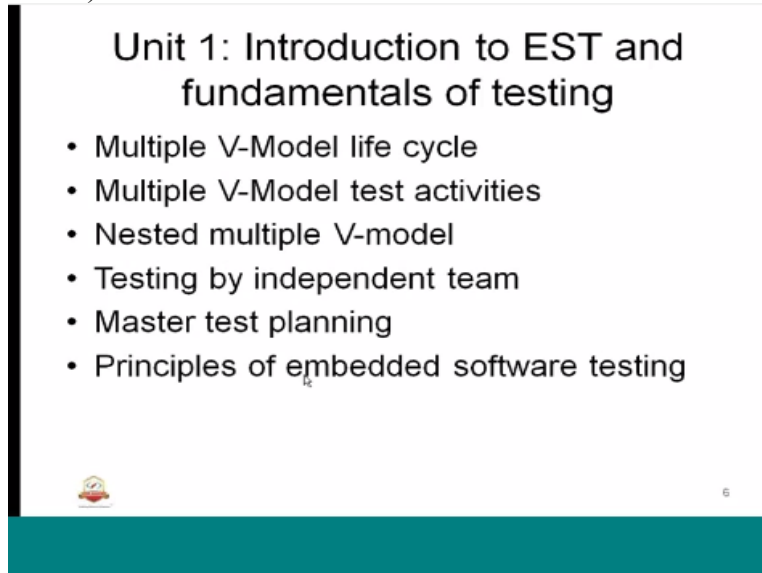


5

Then T-emb method we understood with LITO principles software life cycle entry exit criteria, prototyping lifecycle and example we have gone through. A formal life cycle and example we understood, V-model life cycle mechanism or embedded software development testing we have gone through. And different life cycle processes with an example of entry exit criteria we have studied life cycle process example.

We took consumer electronics example. The different stages of all life process and process through each other and automotive embedded projects testing phrases and process also we studied to understand.

(Refer Slide Time: 32:23)



Unit 1: Introduction to EST and fundamentals of testing

- Multiple V-Model life cycle
- Multiple V-Model test activities
- Nested multiple V-model
- Testing by independent team
- Master test planning
- Principles of embedded software testing

Then multiple V model life cycles based on that famous embedded software testing book. We referred and understood. The same way we also went through the V-model, multiple V-model test activities. Nested multiple V-model, testing by an independent team, master test planning, principles of embedded software testing about I think ten principles we have studied about embedded software testing which is highlighted in one of the embedded book. That

(Refer Slide Time: 33:00)

Unit 2: Testing Methods

- Dynamic Testing
- Static vs dynamic testing
- Dynamic testing types
 - Black box techniques
 - White box techniques
- Testing technique strategy general
- Test case selection methods
- Black box and white box testing coverage aspects, adv & disadv



7

We came to the conclusion about the embedded software testing MS of system, basics setup planning and all that. Next we started studied on the unit 2 in detail of embedded software testing of testing methods and testing methods have the various chapters various sessions about the dynamic testing, static versus dynamic testing context, dynamic testing types, black box, and white box techniques.

Testing techniques strategy in general and in specific is depending on the complexity and the nature of the embedded software testing and the test case selection methods, black box and white box testing coverage aspects advantages and disadvantages. Black box testing (Refer Slide Time: 33:52)

Unit 2: Testing Methods

- Black box testing
 - Equivalence Partitioning
 - Boundary value analysis
 - State or event transition
- Black box testing examples, valid and invalid equivalence class
- Boundary value analysis, applicability, examples
- Exercise with temperature sensor testing



8

How to carry the black box testing and how to draw various test cases in selection and in that way we studied that the fundamentals and basics about the lack box testing in equivalence partition, boundary value analysis, state transition or event transition. So these primary methods of black box testing we had gone through. Black box testing examples valid a invalid equivalence classes.

Boundary values analysis its applicability examples and I think we took the temperature sensor example with high and low values. So, what are the values we can feed? How you can use the techniques in terms of black box testing using boundary value analysis equivalence classes, coverage aspects also. State transition testing
(Refer Slide Time: 34:47)

Unit 2: Testing Methods

- State transition testing
 - Events, actions, activities, states, transitions
- Exercise example with EUI (embedded unit instrument) – 5 op modes
- State transition testing techniques
 - State-event table
 - Transition tree
 - Legal, illegal and guard test cases
- VCR example of state events and transition testing, transition tree



9

I think we had gone through the VCR example simple and the various events, actions, activities, states and transitions all it is going to occur and how we can test with, we had gone through. We had an exercise example with EUI embedded unit instrument, 5 operation modes with the example we gone through try to understand.

And put various testing in the strategy. Tests or a state transition testing techniques state even table we have gone through the example. Transition tree legal illegal and guard test cases for that test events. VCR examples of state events, transition testing and transition tree drawn and gone through.

(Refer Slide Time: 35:45)

Unit 2: Testing Methods

- Model based Design (MBD) intro.
- Model based testing(MBT)
- Taxonomy of model based testing
- Dynamic testing – black box and white box testing approach
- Coverage testing – structural coverage testing
- Statement coverage
- Decision or branch coverage
- Condition coverage
- Other types of testing
 - Data flow testing
 - Branch condition combination testing
 - Modified condition testing – MCDC
 - LCSAJ
- SW instrumentation

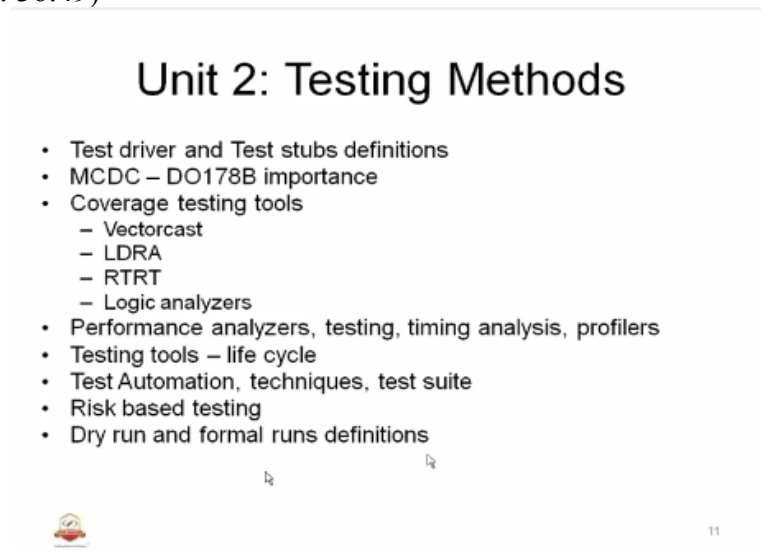


10

The next type of testing method you can model based testing. We understood the basics of model based testing. Model based techniques, taxonomy of model based testing, we use the various instrument for that models such as catalog and all that. And accordingly we are going to draw a strategy for that particular model and we are going to take care of the testing aspects of the model based testing.

Then dynamic testing black box and white box testing, approach coverage testing in terms of structural coverage and statement coverage. Decision and branch coverage, condition coverage other types of testing like data flow testing branch condition combination testing, modified condition testing which is also called as MCDC, LCSAJ its principles then unit testing and software instrumentation. What it means and how they are used with the various tools.

(Refer Slide Time: 36:49)



The slide is titled "Unit 2: Testing Methods" in a large, bold, black font. Below the title is a bulleted list of topics. The list includes: "Test driver and Test stubs definitions", "MCDC – DO178B importance", "Coverage testing tools" (with sub-bullets for "Vectorcast", "LDRA", "RTRT", and "Logic analyzers"), "Performance analyzers, testing, timing analysis, profilers", "Testing tools – life cycle", "Test Automation, techniques, test suite", "Risk based testing", and "Dry run and formal runs definitions". At the bottom left of the slide is a small cartoon character icon, and at the bottom right is the number "11".

- Test driver and Test stubs definitions
- MCDC – DO178B importance
- Coverage testing tools
 - Vectorcast
 - LDRA
 - RTRT
 - Logic analyzers
- Performance analyzers, testing, timing analysis, profilers
- Testing tools – life cycle
- Test Automation, techniques, test suite
- Risk based testing
- Dry run and formal runs definitions

Then test driver and test stubs definitions we had MCDC-DO178B prospective of importance we had gone through few slides. Then coverage testing tools vector cast LDRA, RTRT, logic analyzers. We understood what they do and I think one of the practical sessions we go to one of the possible line from like on of the tool we can use it with an example project. Performance analysis is testing.

Timing analysis profilers take the deviation all that we can develop the test strategy and do the testing or the embedded software testing. Testing tools and life cycle, test automation techniques, test suits, risk based testing we understood and we had definm4ed about dry run and formal runs definitions. The next one is

(Refer Slide Time: 37:51)

Unit 3: Static analysis & code reviews

- Static Testing vs. dynamic testing
- Static analysis
 - Control coupling, data coupling
 - Static metrics analysis
 - McCabe Cyclomatic complexity & examples
 - LOC, Fan-in and Fan-out
 - Nesting levels
- Static analysis tools
 - Understand for c/c++, polyspace, coverity, QAC, Cantata, LDRA testbed..
 - MISR rule checker, PC-Lint, Logiscope rule checker..
 - Call tree analysis, WCET analysis, Stack analysis, stack overflow
 - Coding standards, rules



12

12

Third unit which is about the static analysis and code reviews where we do static testing and analysis reviews inspection and all that and we had gone differences between static testing and dynamic testing both is complimenting each other, both are needed static analysis as control coupling, data coupling, static metrics analysis cyclomatic complexity examples and few examples.

We had gone through with the call tree example generated from the tool and lines of code of the exhibition fan-in fan-out, nesting levels in terms of generating the matrices. Static analysis tools like understand for c++, polyspace, coverity, QAC, Cantata, LDRA tested etc they are used. MISR rule checker, PC-lint. Logic scope rule checker are use for seeing the report of the particular embedded software testing the aspects in terms of code how they are used. Call tree analysis.

WCET analysis, stack analysis, stack overflow and coding standard and rules which are applied in the embedded software development will be checked in the tools and the report added.
(Refer Slide Time: 39:24)

Unit 3: Static analysis & code reviews

- Test Metrics
 - Quantification
 - Test metrics lifecycle
 - Metrics types
 - SW testing metrics
 - Defect acceptance, defect rejection, Execution productivity, Test efficiency, defect severity index, automation coverage, Effort Variance, Schedule Variance, scope change
 - Metrics representation, reporting
 - Trend chart, burn-down chart
 - Metrics capture tool
 - Defect management - Bugzilla



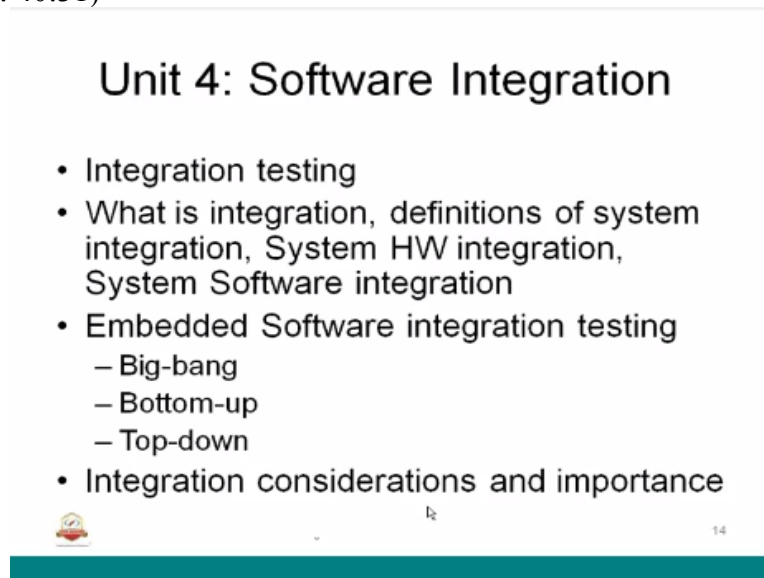
13

13

And test metrics quantification test metrics life cycle, metrics types, software testing metrics of various metrics that are been generated for embedded software testing and defect acceptance, defect rejection, execution productivity, test efficiency, defect severity index, automation coverage, effort variance, schedule variance scope change all these definitions we had metrics representation.



How they are maybe reported on daily bases, weekly bases and trend chart how they are shown in to the higher management customers. Burn-down charts suppose we if sort of activities going to take one month and how are you going to reset one month using the burn-down chart we are going to try it and report. And as we progress we are going to operate it and report it. And metrics capture tool there are waste tools that are used for managing the test aspects, defects aspects and all. Defect management- bugzilla as been used.

(Refer Slide Time: 40:31)



Unit 4: Software Integration

- Integration testing
- What is integration, definitions of system integration, System HW integration, System Software integration
- Embedded Software integration testing
 - Big-bang
 - Bottom-up
 - Top-down
- Integration considerations and importance

  14

The next unit is about software integration, integration testing. This is also an important part of the embedded software testing. We had defined about integration testing. What is integration? Definitions of system integration, system hardware integration, system software integration were we use software logic. In system hardware integration we use hardware and software. Together we are going to build integrate and testing. And types of embedded software integration testing big-bang, bottom-up, top-down and considerations for integration testing and its importance we studied.

(Refer Slide Time: 41:10)

Unit 4: Software Integration

- Integration test strategy
- Comparison of various strategies
- Hybrid integration test strategy
- Centralized integration
- Layer integration, client/server integration, collaboration integration etc..
- Integration test environment
- System Integration test
- Integration from use case perspective, generating test cases



15

Integration test strategy, comparison of various strategies like top-down, bottom-up, big-bang and all that we had gone through. We had a comparison about it then hybrid integration test strategy which commence both top-down as well as bottom-up integration test approach. Then we have a centralized integration, layer integration, client/server integration, collaboration integration etc.

Based on the type of nature of the project we are going to decide and do the software integration testing. Then integration testing environment I will fetch like system integration test, integration from use case prospective they are specifically used in EML type of projects and we use that use case to generate the test cases. Then we had gone

(Refer Slide Time: 42:05)

Unit 4: Software Integration

- Regression testing
 - Re-testing, definition of regression testing
 - Test strategy, testing areas
 - Testing automation
 - Change analysis and relative importance
 - Use case example
 - Prioritizing regression test cases
- Test case maintenance
- Build process



16

Through regression testing which is re-testing and regression testing, definition of regression testing, test strategy, test areas, testing automation in terms of batch execution and processing. Change analysis and relative importance. Use case example we have seen. Prioritizing regression

test cases and what is the basis for doing that priority and all that test case maintenance because very important.

Because once I made software testing basic testing is done. It is gone to update or modify for a period and how are you going to maintain it, the project. What are the steps that are involved for maintain what are the test cases we are going to exercise for the future versions of different embedded software releases. And build process we had gone through
(Refer Slide Time: 43:05)

Unit 5: Test Management

- Configuration Management
- Test Management
- Configuration management elements
- CM process
- SCM (Software Configuration Management)
- SCM activities
 - Planning, SCI, control , status accounting, software configure audit
- SCM process example and tasks
- CC and SCB
- Configuration items guidelines/lifecycle



And last unit is about test management which we are concluded in today's session. It talks about configuration management, test management, configuration management elements, CM process, SCM it is software configuration management. Software configuration management activities such as planning, software configuration index, control, status accounting and software configuration audit.

These are different activities that are used for resume the CM. SCM process example and tasks, configuration control and SCB software configuration board, controller board. Configuration items guidelines/lifecycle also we have gone through and
(Refer Slide Time: 43:58)

Unit 5: Test Management

- SCM phases
 - Initiation
 - Planning
 - Execution
 - Closure
- Version control
- Baseline
- Workspace management
- Change Management
- Incident Management
- CM tools – PVCS, SVN, MKS...



18

SCM phases like initiation, planning, execution, closure. So, we had through one of the session and important thing is about the version control or the revision control and how are you going to do the base line of the auto cuts? Workspace management comes of the repository How that is been used by developing testing tool of the configuration control people that mean basically manages that and change management, incident management as per the session we had gone through. Last part was about CM tools.

(Refer slide Time: 44:32)

Unit 5: Test Management

- Test Management
- Test process in relation with software V-model
- Testing – design by contract
- Agile development process
- Agile development scrum process
- Agile testing scrum process
- Test driven development
- Test management and control
- Test management tool
- Defect management tool



19

PVCS, SVN, MKS and test management aspects we had studied in terms of testing process which is in relation with software V-model designed by contract, part of the testing, agile development process, agile development scrum process, agile testing scrum process, test driven development test management and control, test management tool, defect management tool. So, these are the part of the test management.

So, that is about test management in unit 5. So, you know 5 units of embedded software testing which we had gone through. With that we will come to the conclusion of all the units of the

embedded software testing. So, in the next four or five session we will go to the practical aspects of embedded software testing we will divide accordingly the various element that are used for embedded software testing in terms of practicality. Okay.