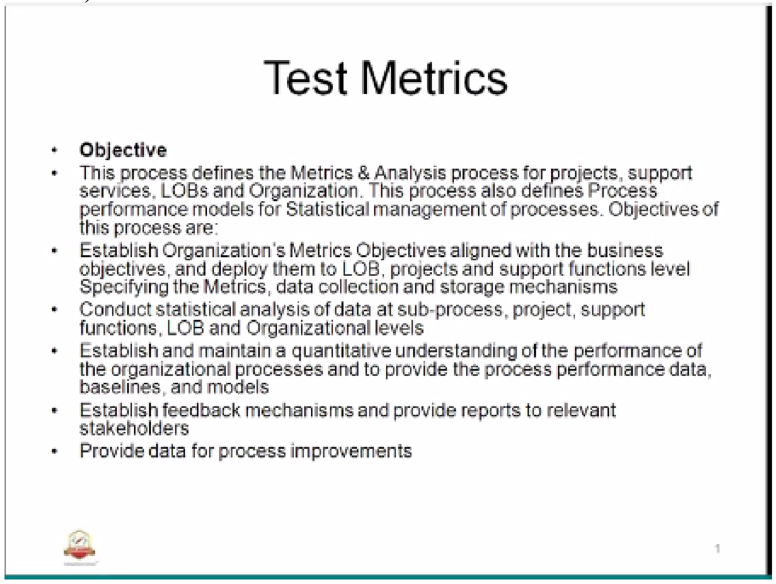
Embedded Software Testing
Unit 3: Static analysis and code reviews & Metrics
Lecture 5

Seer Akademi-NPTEL MOU

Hi all welcome to the next session of our embedded software testing that unit 3 series and the lecture 5 and unit 3: this is about static analysis and code review & metrics, and reveal more and study more on the metrics.

(Refer Slide Time: 00:26)



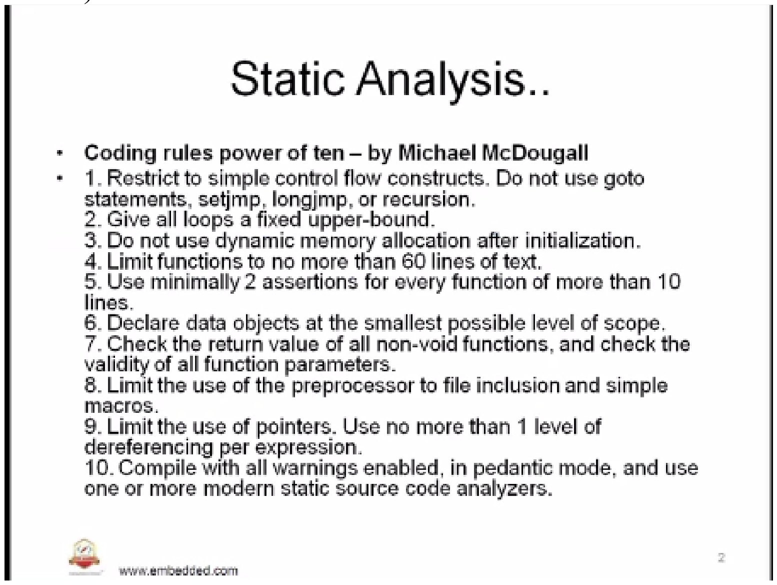
Test Metrics

- **Objective**
- This process defines the Metrics & Analysis process for projects, support services, LOBs and Organization. This process also defines Process performance models for Statistical management of processes. Objectives of this process are:
- Establish Organization's Metrics Objectives aligned with the business objectives, and deploy them to LOB, projects and support functions level Specifying the Metrics, data collection and storage mechanisms
- Conduct statistical analysis of data at sub-process, project, support functions, LOB and Organizational levels
- Establish and maintain a quantitative understanding of the performance of the organizational processes and to provide the process performance data, baselines, and models
- Establish feedback mechanisms and provide reports to relevant stakeholders
- Provide data for process improvements

1

Also we will try to recap what we studied in the last class.

(Refer Slide Time: 00:35)



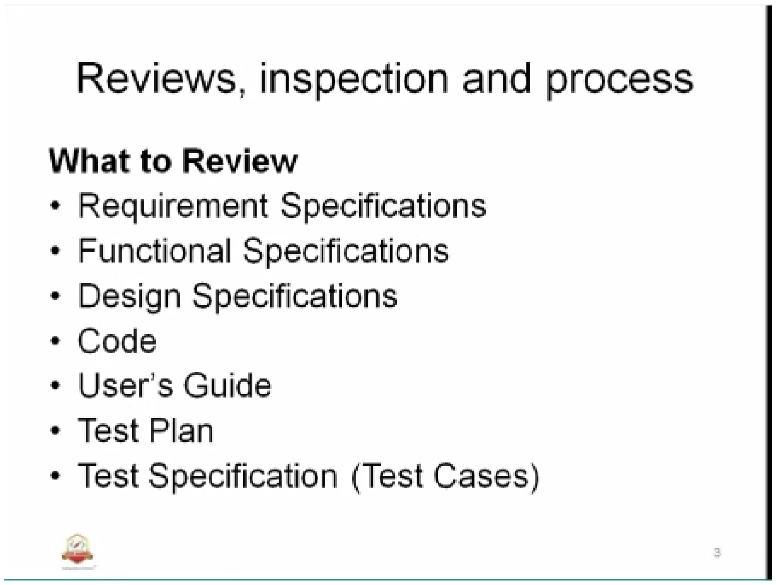
Static Analysis..

- **Coding rules power of ten – by Michael McDougall**
- 1. Restrict to simple control flow constructs. Do not use goto statements, setjmp, longjmp, or recursion.
  2. Give all loops a fixed upper-bound.
  3. Do not use dynamic memory allocation after initialization.
  4. Limit functions to no more than 60 lines of text.
  5. Use minimally 2 assertions for every function of more than 10 lines.
  6. Declare data objects at the smallest possible level of scope.
  7. Check the return value of all non-void functions, and check the validity of all function parameters.
  8. Limit the use of the preprocessor to file inclusion and simple macros.
  9. Limit the use of pointers. Use no more than 1 level of dereferencing per expression.
  10. Compile with all warnings enabled, in pedantic mode, and use one or more modern static source code analyzers.

www.embedded.com 2

So we go on throw the coding rules, this is one of the important static annalistic. There are 10 golden rules by Michael McDougall, so those need to be up to time and reviewed and unleashed the embedded software code.

(Refer Slide Time: 00:54)



Reviews, inspection and process

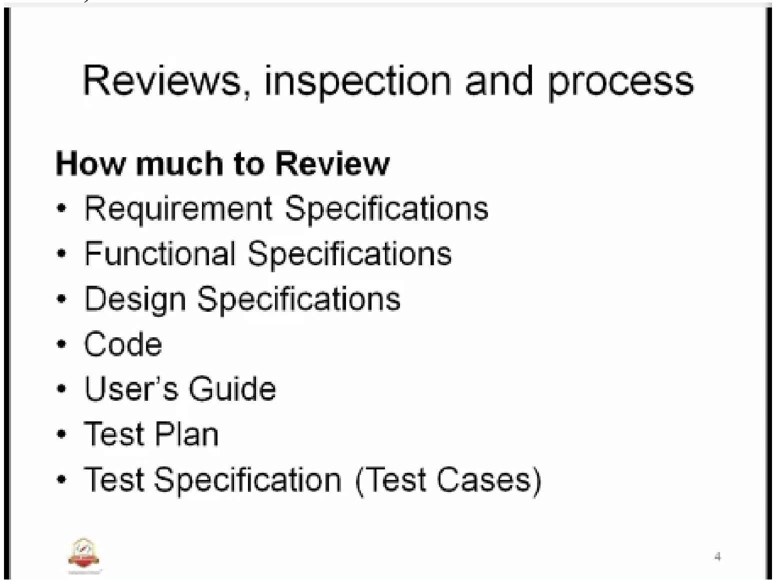
What to Review

- Requirement Specifications
- Functional Specifications
- Design Specifications
- Code
- User's Guide
- Test Plan
- Test Specification (Test Cases)

3

And it is also important to know what should we be reviewing it, and at what stage. So all these will be as per the plan, again is what we need to review, there are guidelines and rules and those rules are all against the inch of the these artifacts in the embedded software life cycle, so requirements, function specification, design specification have their own guidelines and rules, and checkers also will be different and these life cycles we will have a transition criteria in the terms of re-exit. So the transition criteria is need to be satisfied in order to, say that product is ready for going to the next to the life cycle output.

(Refer Slide Time: 01:44)



Reviews, inspection and process

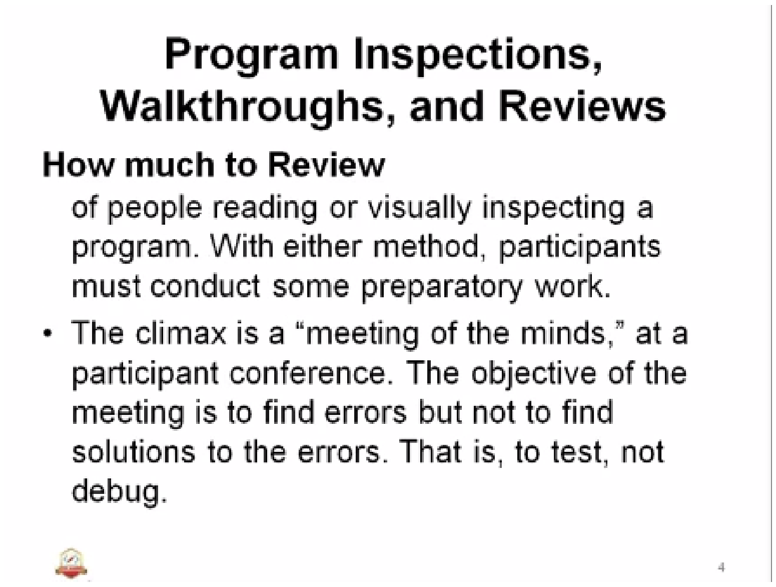
How much to Review

- Requirement Specifications
- Functional Specifications
- Design Specifications
- Code
- User's Guide
- Test Plan
- Test Specification (Test Cases)

4

So we have how much to review also is one of the product term, so there are different reviewing a body like here level, self review, offline inspection, likewise there are many types of reviews, and we will have software guidelines and rules, that is aims objective depending on the complexity of the embedded software system. So that we will decide how much we need to review.

(Refer Slide Time: 02:14)



Program Inspections, Walkthroughs, and Reviews

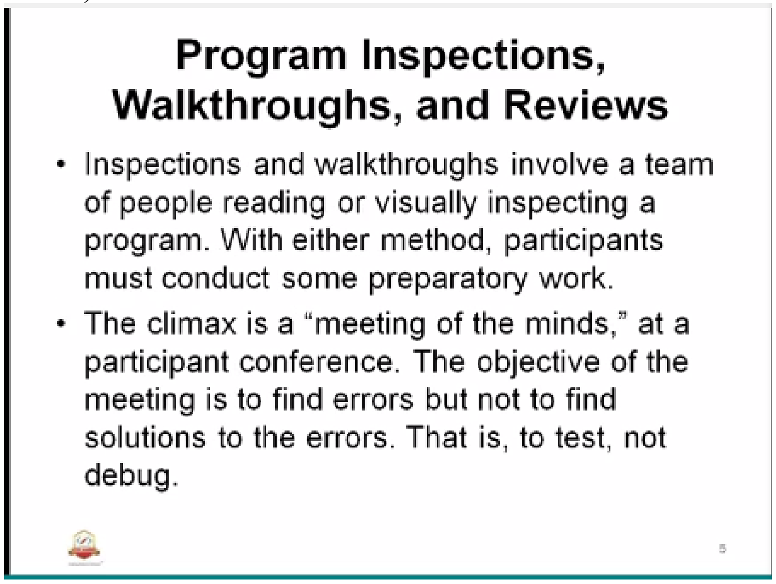
How much to Review

of people reading or visually inspecting a program. With either method, participants must conduct some preparatory work.

- The climax is a "meeting of the minds," at a participant conference. The objective of the meeting is to find errors but not to find solutions to the errors. That is, to test, not debug.

4

(Refer Slide Time: 02:15)



Program Inspections, Walkthroughs, and Reviews

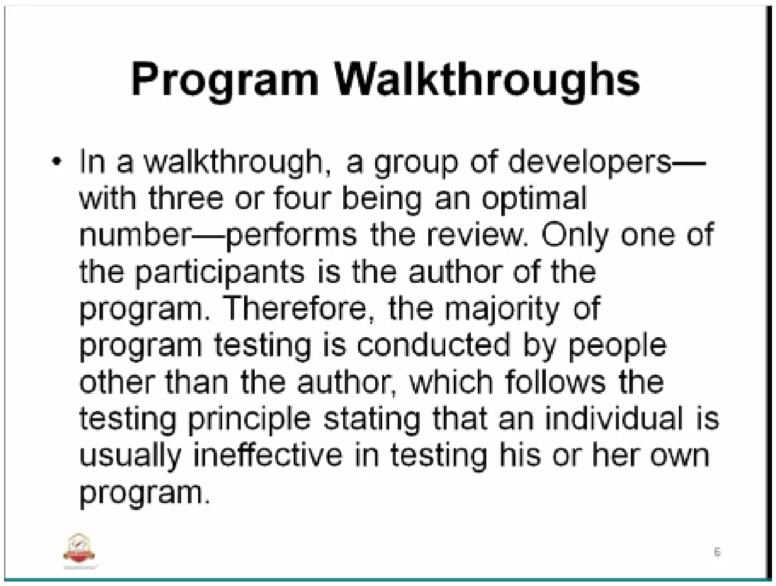
- Inspections and walkthroughs involve a team of people reading or visually inspecting a program. With either method, participants must conduct some preparatory work.
- The climax is a "meeting of the minds," at a participant conference. The objective of the meeting is to find errors but not to find solutions to the errors. That is, to test, not debug.

5

Next one we studied about these program inspections, walkthroughs, and reviews. Basically there will be a team involved in while doing the program inspection they will offset will be given, set of documents to read and visually instruct before the meeting and the participants will gather in the conference, meeting, the participant can be choose geographically who work, that's what we see in the industry.

So basically they meet each other, the objective of meeting is to find hour, but they do not want to bother about the solutioning of the errors. That is how to debug and how to fix that in all work and it is not part of this reviews, it just identifies the errors.

(Refer Slide Time: 03:09)

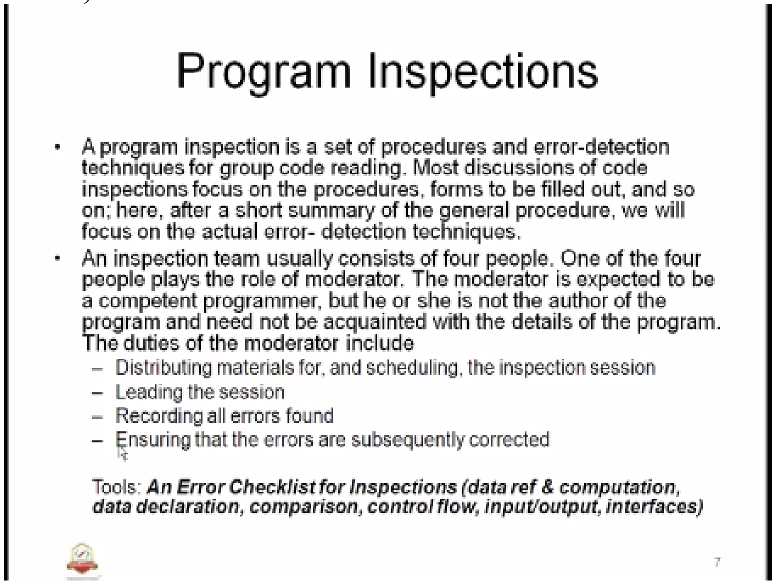

Program Walkthroughs

- In a walkthrough, a group of developers—with three or four being an optimal number—performs the review. Only one of the participants is the author of the program. Therefore, the majority of program testing is conducted by people other than the author, which follows the testing principle stating that an individual is usually ineffective in testing his or her own program.

6

So In walkthrough typically a group of developers with three are four being in the optimal number will perform the review against the subjective depending on the complexity. So one participant in this it will be an author, so others will review again as projectors and have conclusion error and in concern with the author of the program.

(Refer Slide Time: 03:35)


Program Inspections

- A program inspection is a set of procedures and error-detection techniques for group code reading. Most discussions of code inspections focus on the procedures, forms to be filled out, and so on; here, after a short summary of the general procedure, we will focus on the actual error- detection techniques.
- An inspection team usually consists of four people. One of the four people plays the role of moderator. The moderator is expected to be a competent programmer, but he or she is not the author of the program and need not be acquainted with the details of the program. The duties of the moderator include
  - Distributing materials for, and scheduling, the inspection session
  - Leading the session
  - Recording all errors found
  - Ensuring that the errors are subsequently corrected

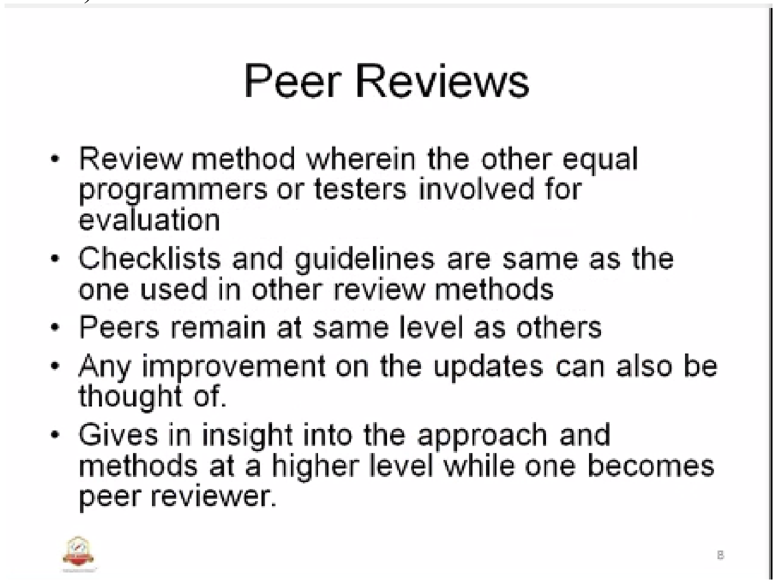
  Tools: ***An Error Checklist for Inspections (data ref & computation, data declaration, comparison, control flow, input/output, interfaces)***

7

So there is a group code reading that is interior code set will be divided in to two group, and that group will go through the code and they will discuss about the code issues and inspections whatever they have done in the terms of procedures and all that, so they will flirt a checklist, checklist is also called as a tool, so what the checklist will have is a inspections in the terms of declarations comparing events, controls flow, the data flow, the input output of the program interfaces all this tend to be checked basically.

That's what they do, and when they do a inspection on, there will be typically four to five people, one of them will be called as what? Moderator, so these were primarily responsible for all the activities and he also has to ensure the errors that are being corrected, I guess what is been found as well in the conference meeting.

(Refer Slide Time: 04:51)



Peer Reviews

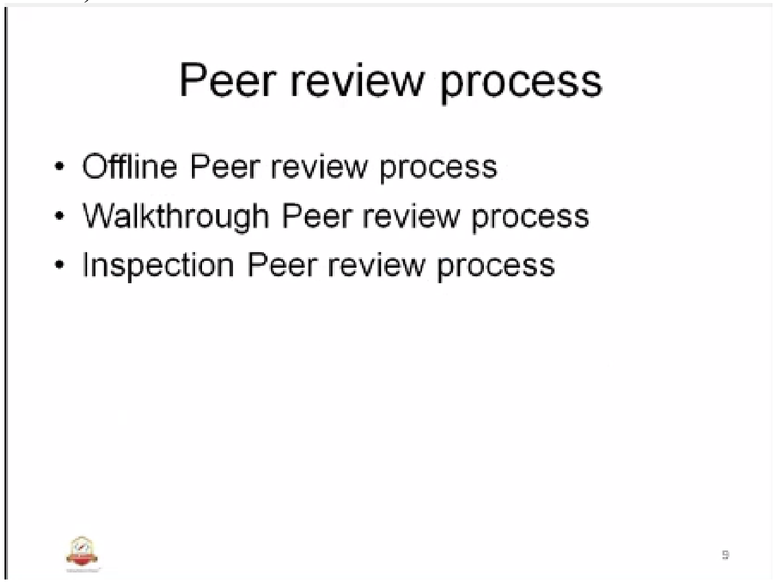
- Review method wherein the other equal programmers or testers involved for evaluation
- Checklists and guidelines are same as the one used in other review methods
- Peers remain at same level as others
- Any improvement on the updates can also be thought of.
- Gives in insight into the approach and methods at a higher level while one becomes peer reviewer.

8

Peer review is something like he method, wherein the other equal programmers or testers involved for evaluation. That means, group of people are at the same level in the terms of development or testing against the each other artifacts they will do a review, and off course here also checklist and guidelines are same as they used in the other review methods. Any improvement also can be followed, so all the updates optimization lecture by peers, that will be taken care as they bugs fixed along with the improvement.

So it also gives an insight into approach and methods that I have in the higher level from the detection clear reviewer.

(Refer Slide Time: 05:32)



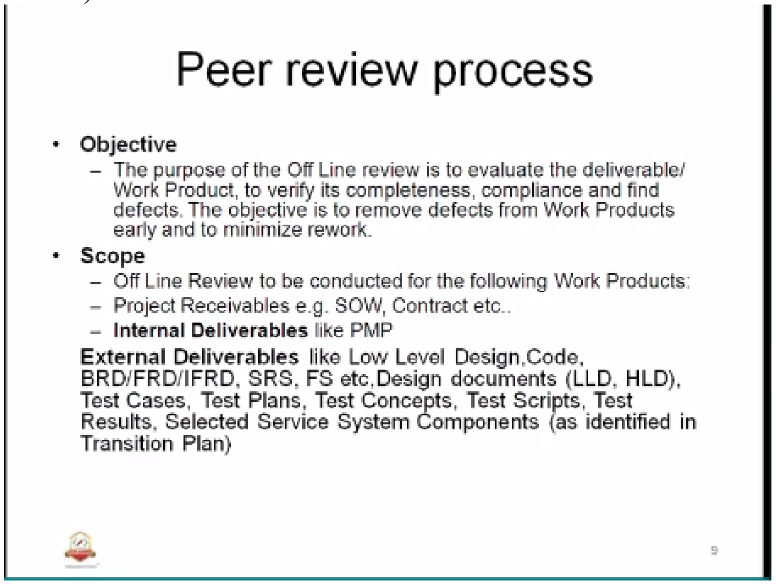
Peer review process

- Offline Peer review process
- Walkthrough Peer review process
- Inspection Peer review process

9

So the types of peer is, review process what we studied in the offline peer review, walkthrough peer review, inspection peer review.

(Refer Slide Time: 05:44)



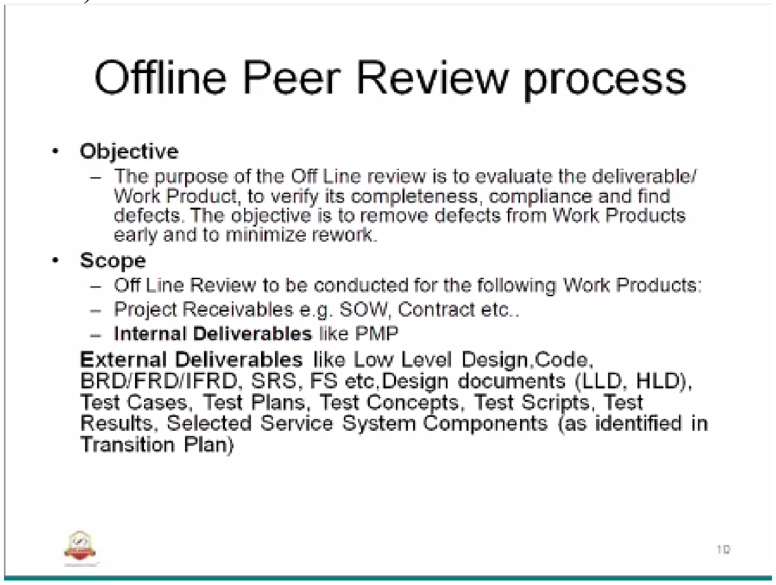
Peer review process

- **Objective**
  - The purpose of the Off Line review is to evaluate the deliverable/ Work Product, to verify its completeness, compliance and find defects. The objective is to remove defects from Work Products early and to minimize rework.
- **Scope**
  - Off Line Review to be conducted for the following Work Products:
  - Project Receivables e.g. SOW, Contract etc..
  - **Internal Deliverables** like PMP

  External Deliverables like Low Level Design,Code, BRD/FRD/IFRD, SRS, FS etc,Design documents (LLD, HLD), Test Cases, Test Plans, Test Concepts, Test Scripts, Test Results, Selected Service System Components (as identified in Transition Plan)

9

(Refer Slide Time: 05:44)



Offline Peer Review process

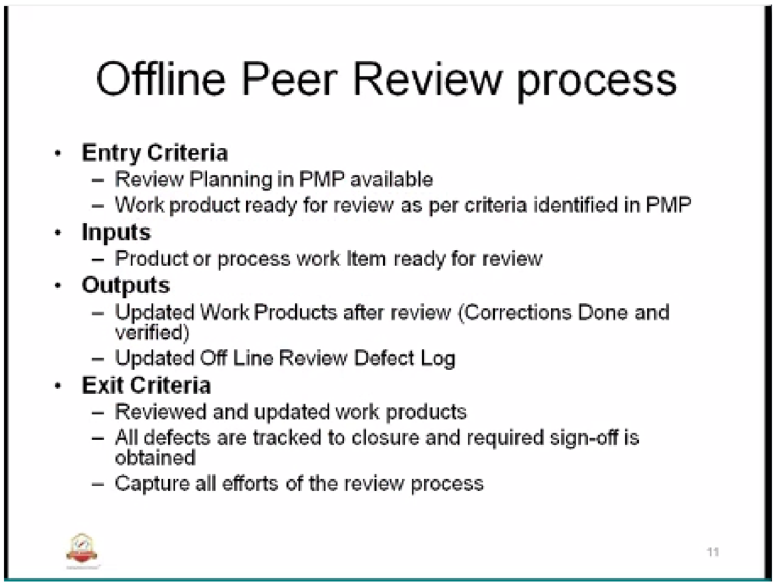
- **Objective**
  - The purpose of the Off Line review is to evaluate the deliverable/ Work Product, to verify its completeness, compliance and find defects. The objective is to remove defects from Work Products early and to minimize rework.
- **Scope**
  - Off Line Review to be conducted for the following Work Products:
  - Project Receivables e.g. SOW, Contract etc..
  - **Internal Deliverables** like PMP

  External Deliverables like Low Level Design,Code, BRD/FRD/IFRD, SRS, FS etc,Design documents (LLD, HLD), Test Cases, Test Plans, Test Concepts, Test Scripts, Test Results, Selected Service System Components (as identified in Transition Plan)

10

So basically the objective of always will be to evaluate the deliverable work product, to verify the completeness, compliance, and find defects. Scope is to conduct the offline review of the different products, it is the example I have put, it could be anything of the embedded software life cycle of the product management plan document are the test static files etc. and the deliverables is a part of the offline peers, lower level designs, test cases test plans anything that can perform for this offline peer review.

(Refer Slide Time: 06:26)

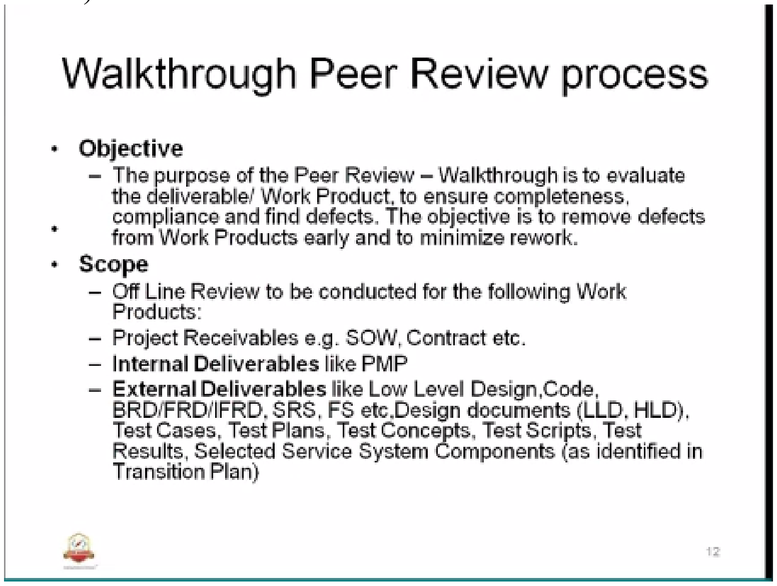


Offline Peer Review process

- **Entry Criteria**
  - Review Planning in PMP available
  - Work product ready for review as per criteria identified in PMP
- **Inputs**
  - Product or process work Item ready for review
- **Outputs**
  - Updated Work Products after review (Corrections Done and verified)
  - Updated Off Line Review Defect Log
- **Exit Criteria**
  - Reviewed and updated work products
  - All defects are tracked to closure and required sign-off is obtained
  - Capture all efforts of the review process

11

And there is a four important of processor aspects. Entry criteria inputs outputs, and the exit criteria each has its own transition mechanism in terms of deliverables of the outputs.
(Refer Slide Time: 06:44)

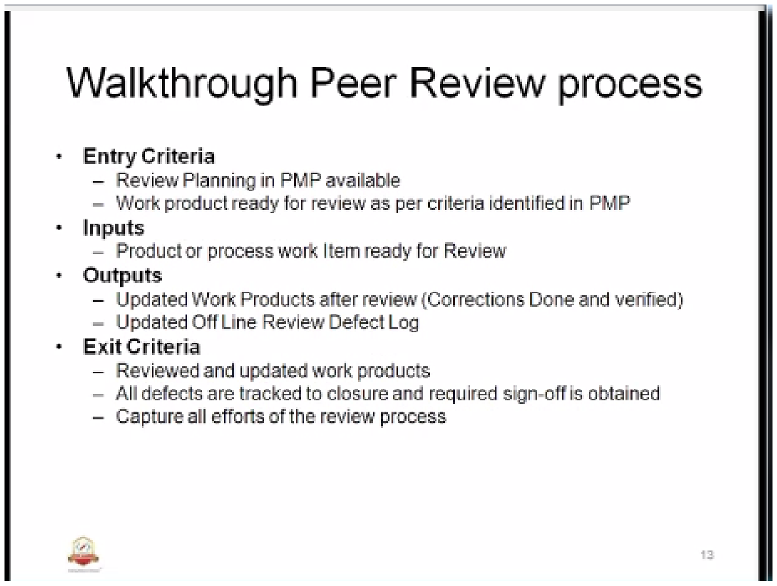


Walkthrough Peer Review process

- **Objective**
  - The purpose of the Peer Review – Walkthrough is to evaluate the deliverable/ Work Product, to ensure completeness, compliance and find defects. The objective is to remove defects from Work Products early and to minimize rework.
- **Scope**
  - Off Line Review to be conducted for the following Work Products:
  - Project Receivables e.g. SOW, Contract etc.
  - **Internal Deliverables** like PMP
  - **External Deliverables** like Low Level Design,Code, BRD/FRD/IFRD, SRS, FS etc,Design documents (LLD, HLD), Test Cases, Test Plans, Test Concepts, Test Scripts, Test Results, Selected Service System Components (as identified in Transition Plan)

12

So next type of peer review is walkthrough, the objectives, scope,
(Refer Slide Time: 06:49)



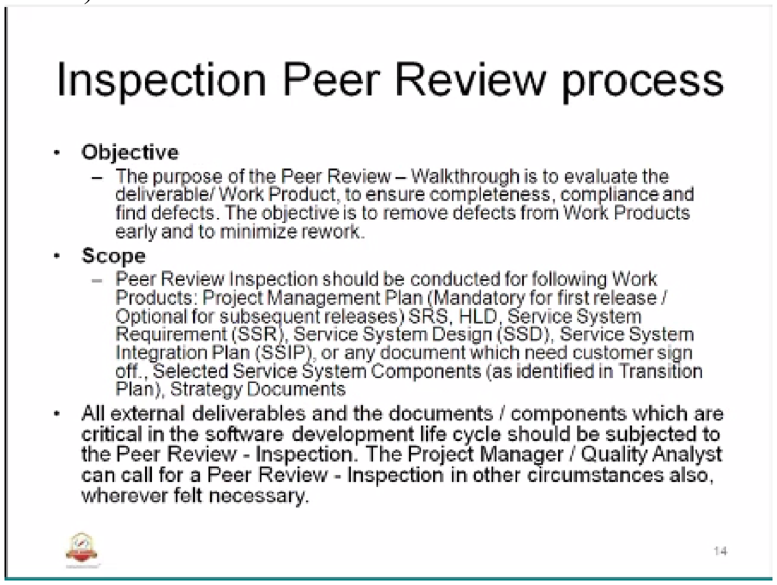
Walkthrough Peer Review process

- **Entry Criteria**
  - Review Planning in PMP available
  - Work product ready for review as per criteria identified in PMP
- **Inputs**
  - Product or process work Item ready for Review
- **Outputs**
  - Updated Work Products after review (Corrections Done and verified)
  - Updated Off Line Review Defect Log
- **Exit Criteria**
  - Reviewed and updated work products
  - All defects are tracked to closure and required sign-off is obtained
  - Capture all efforts of the review process

13

Entry input and output executed in our basically removed outside, that is needed to be identified for this. So next one is inspection peer review,

(Refer Slide Time: 06:59)



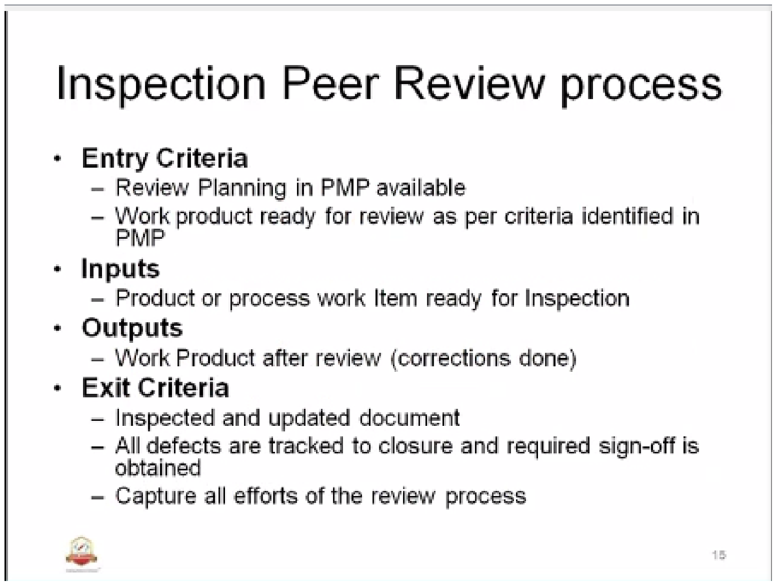
Inspection Peer Review process

- **Objective**
  - The purpose of the Peer Review – Walkthrough is to evaluate the deliverable/ Work Product, to ensure completeness, compliance and find defects. The objective is to remove defects from Work Products early and to minimize rework.
- **Scope**
  - Peer Review Inspection should be conducted for following Work Products: Project Management Plan (Mandatory for first release / Optional for subsequent releases) SRS, HLD, Service System Requirement (SSR), Service System Design (SSD), Service System Integration Plan (SSIP), or any document which need customer sign off., Selected Service System Components (as identified in Transition Plan), Strategy Documents
- All external deliverables and the documents / components which are critical in the software development life cycle should be subjected to the Peer Review - Inspection. The Project Manager / Quality Analyst can call for a Peer Review - Inspection in other circumstances also, wherever felt necessary.

14

Where referred in the earlier slide, a group of people will gather, they go through the artifacts in the terms of big programs code the SRS, HLD, LLD, all together and try to come up with the errors that are there in the executing program also suggest to improve in the terms of efficiency of the program.

(Refer Slide Time: 07:25)



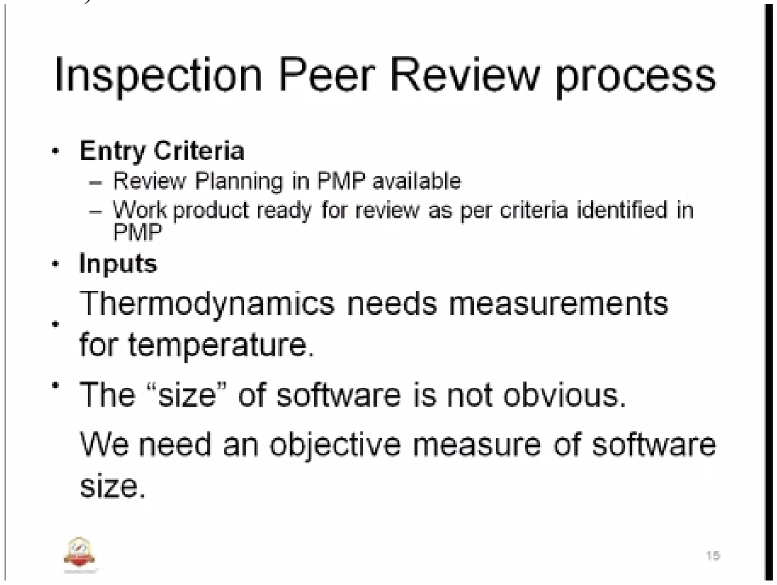
Inspection Peer Review process

- **Entry Criteria**
  - Review Planning in PMP available
  - Work product ready for review as per criteria identified in PMP
- **Inputs**
  - Product or process work Item ready for Inspection
- **Outputs**
  - Work Product after review (corrections done)
- **Exit Criteria**
  - Inspected and updated document
  - All defects are tracked to closure and required sign-off is obtained
  - Capture all efforts of the review process

15

So these also have entry input and output that they given in.

(Refer Slide Time: 07:28)



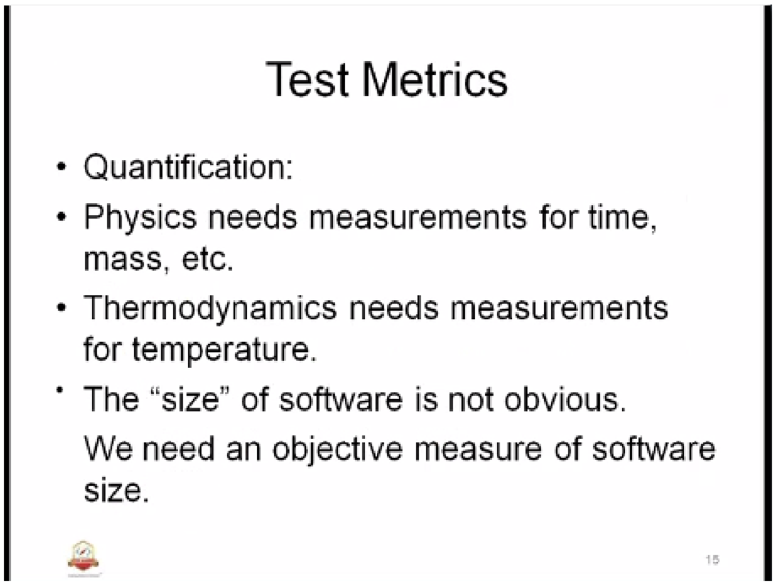
Inspection Peer Review process

- **Entry Criteria**
  - Review Planning in PMP available
  - Work product ready for review as per criteria identified in PMP
- **Inputs**
- Thermodynamics needs measurements for temperature.
- The "size" of software is not obvious.

  We need an objective measure of software size.

15

(Refer Slide Time: 07:29)

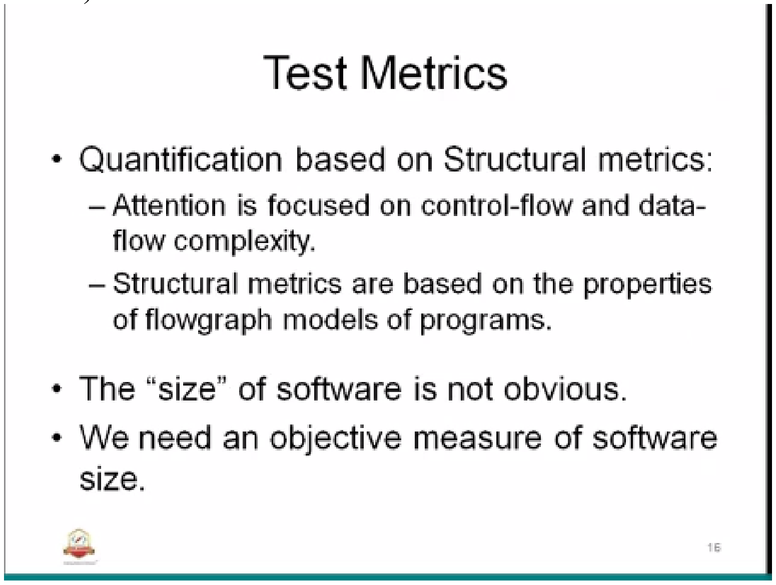

Test Metrics

- Quantification:
- Physics needs measurements for time, mass, etc.
- Thermodynamics needs measurements for temperature.
- The "size" of software is not obvious.

  We need an objective measure of software size.

15

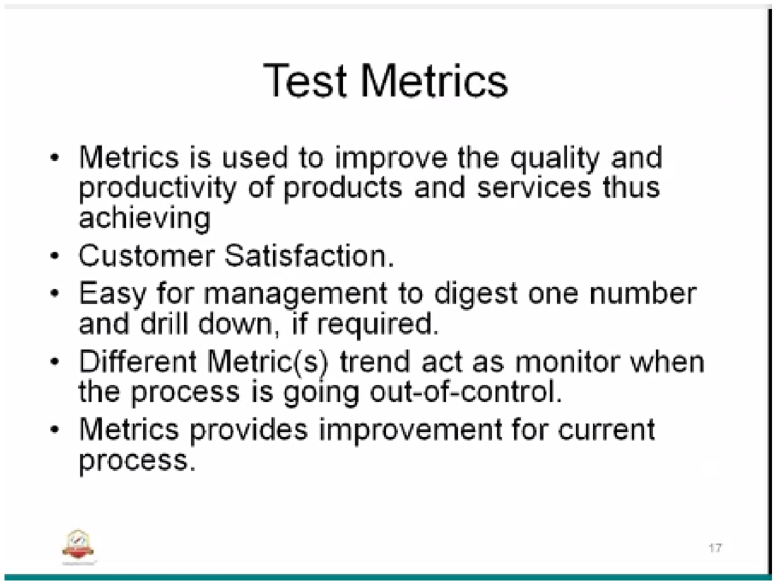
That if we move to the next one test metrics,
(Refer Slide Time: 07:35)


Test Metrics

- Quantification based on Structural metrics:
  - Attention is focused on control-flow and data-flow complexity.
  - Structural metrics are based on the properties of flowgraph models of programs.
- The "size" of software is not obvious.
- We need an objective measure of software size.

16

Test metrics, basically test metrics is not just the launch of code,
(Refer Slide Time: 07:41)



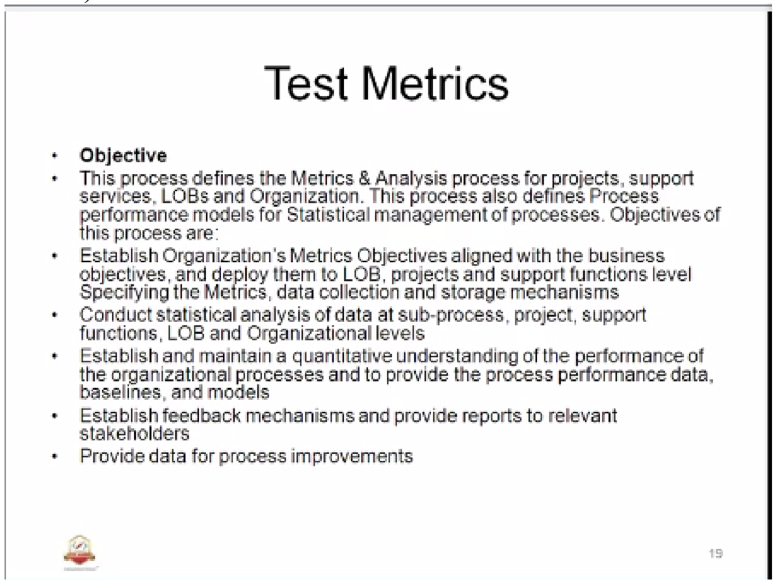
Test Metrics

- Metrics is used to improve the quality and productivity of products and services thus achieving
- Customer Satisfaction.
- Easy for management to digest one number and drill down, if required.
- Different Metric(s) trend act as monitor when the process is going out-of-control.
- Metrics provides improvement for current process.

17

It is basically based on the complexity. And the test how it is going to be conducted, and how it is going to be reported, basically customer how is going to be satisfied all these artifact need to be collated and identified in terms of its importance, to drive what is the metric that is good for a particular embedded system.

(Refer Slide Time: 08:04)



Test Metrics

- **Objective**
- This process defines the Metrics & Analysis process for projects, support services, LOBs and Organization. This process also defines Process performance models for Statistical management of processes. Objectives of this process are:
- Establish Organization's Metrics Objectives aligned with the business objectives, and deploy them to LOB, projects and support functions level Specifying the Metrics, data collection and storage mechanisms
- Conduct statistical analysis of data at sub-process, project, support functions, LOB and Organizational levels
- Establish and maintain a quantitative understanding of the performance of the organizational processes and to provide the process performance data, baselines, and models
- Establish feedback mechanisms and provide reports to relevant stakeholders
- Provide data for process improvements

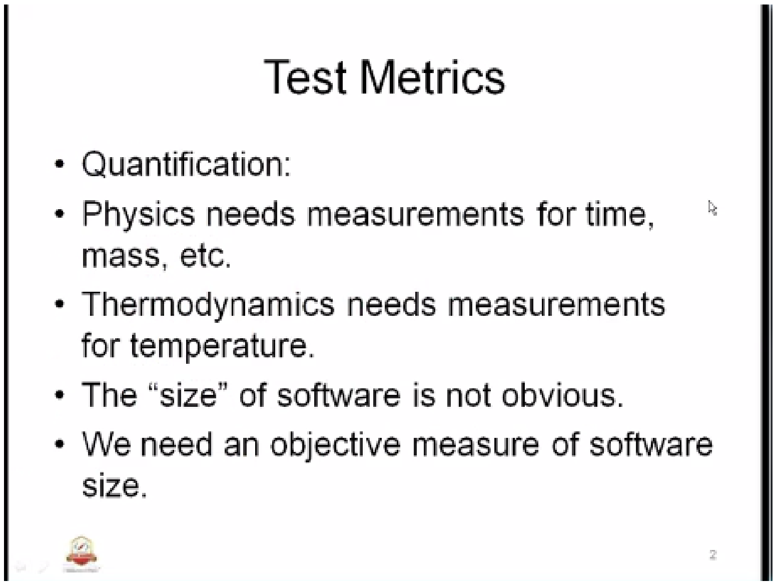
19

Okay, so we will move to the next session with that background, so I will start with the test metric. Here the object is the process defines the metric and analysis process for the projects, support services, LOBs and organization. This defines it process; this process also defines the performance models for statistical management of process. Objectives of this process are, established organization metric objectives, aligned with the business objectives that employ them to the line of business.

The projects and the support functions level specifying the metrics data collections for the mechanism, so basically it is to define the or establish the various metric that organization needs for that particular environment project or the in made project that is falling within the LOB, that

is line of business and it is typically called in the embedded industries, that could be aerospace, the automotive or it could be a such systems part of the main domain.
So the conductor statically analyses of the data that sub posses project, support functions, and LOB organization levels, here sub process are something like, which are derived process from the main process.
Projects you know project and the support function something like we have a project content, project need support functions in the terms of administrative, or resources, resource skills, hr could be involved anything, all these matters in the terms of dependency or radicality for the successful testing program, testing project I would say.
The embedded system projects are also have the life cycle of testing projects, so that is very important to have, all these organizational aspects in terms of processor projector support functions, LOB organizational levels where it is going to make some k decisions, to support smooth functioning of the testing.

So all this will be artifact in the test metric, so that is one of the organization objective, then the next one is established and maintain the connotative understanding of weak performance of the organizational process and provide the process performance data base line and models. Establish feedback mechanism and provides report to relevance stakeholders, that means for each of the process that we have established they should be a good feedback mechanism, that will help in the terms of correcting any issues that itself can have, that means we have a review process and review process itself has some issues, some corrections are required.
So for that how we are going to evaluate based on the feedback, the feedback will bring out some of the improvements in the particular case and particular process that is been followed. So that will be reported to the relevance stakeholder such as test manager or project manager it test relate is the main stakeholder. So there are subjective provide data for processing improvement, so this is not only enough to have , or to identifying the , this mechanism in terms of metric it is also important to have some sort of data which can help in growing the project process, so that is also one of the important test metric.
(Refer Slide Time: 12:53)



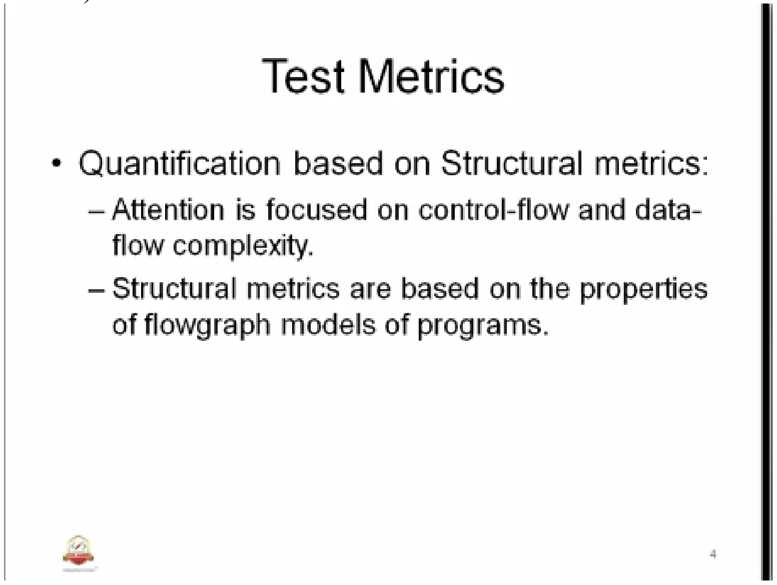
Test Metrics

- Quantification:
- Physics needs measurements for time, mass, etc.
- Thermodynamics needs measurements for temperature.
- The “size” of software is not obvious.
- We need an objective measure of software size.

2

Okay, so test metric again I will go through that and it is very important quantification. So quantification of the data is nothing but the, the metric, but the metric is not just the life’s of collides, life’s of collides is also one of the metric, here reliance of collides size of software is not based on this, life’s of code, size should be subjective based on the complexity,

(Refer Slide Time: 13:21)



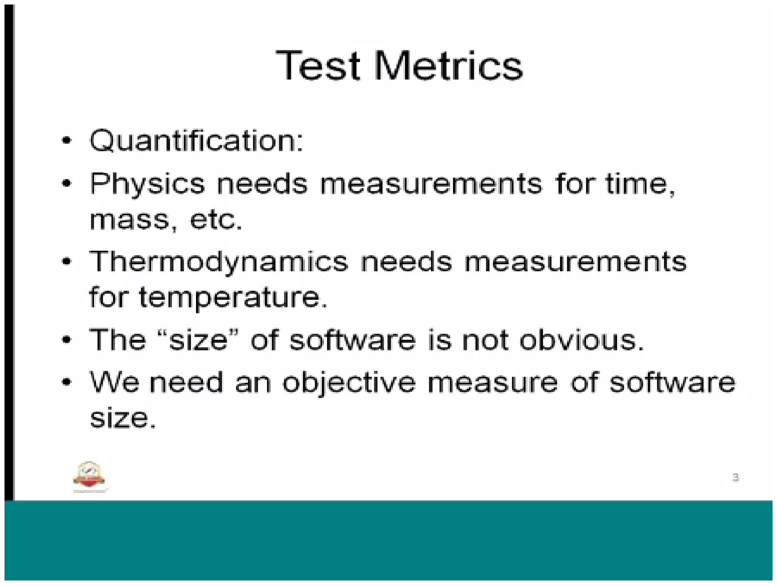
Test Metrics

- Quantification based on Structural metrics:
  - Attention is focused on control-flow and data-flow complexity.
  - Structural metrics are based on the properties of flowgraph models of programs.

4

Structure of the program that is being used. So, basically attention is focused on control-flow and data-flow complexity. Structural metrics are based on the properties of flow graph models of the program. It is very important to have the complexity of the embedded system program and embedded system testing aspects.

(Refer Slide Time: 13:44)



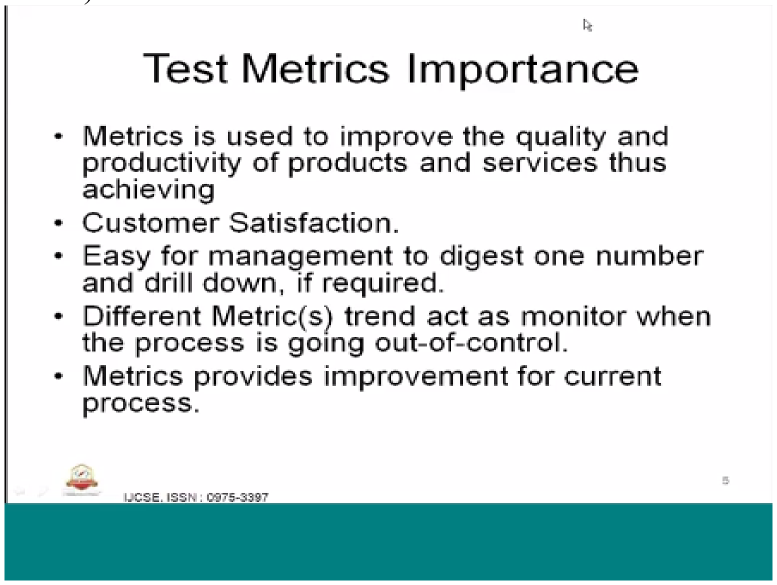
Test Metrics

- Quantification:
- Physics needs measurements for time, mass, etc.
- Thermodynamics needs measurements for temperature.
- The "size" of software is not obvious.
- We need an objective measure of software size.

3

With the help of that the size of the actual metrics that is going to be evaluated in the reported will be done.

(Refer Slide Time: 13:56)



Test Metrics Importance

- Metrics is used to improve the quality and productivity of products and services thus achieving
- Customer Satisfaction.
- Easy for management to digest one number and drill down, if required.
- Different Metric(s) trend act as monitor when the process is going out-of-control.
- Metrics provides improvement for current process.

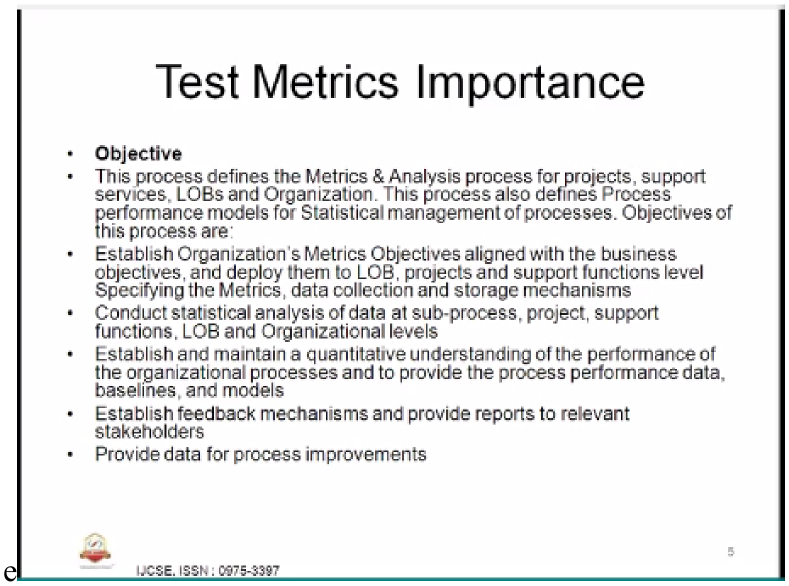
IJCSE, ISSN : 0975-3397

5

So, test, metrics importance you know that very important to have metrics reported to type customer and how much bugs have been fixed? How many are passing how many are failed? How those fails are addressed? How they are going to be justified? How they are fixed in different cases? So, what is the action taken? And what is the trend? So trend we will see in the next slide today. So, basically the trend will give a clear picture of where the testing will be going ahead. So, that is what it gives, the picture of the program.

So, very important to have a defect metrics trend as the monitoring aspect of the test metric, also it provides metric to provide an improvement for the current process, in terms of how much we can optimize? How much you can improve etc.

(Refer Slide Time: 14:47)



Test Metrics Importance

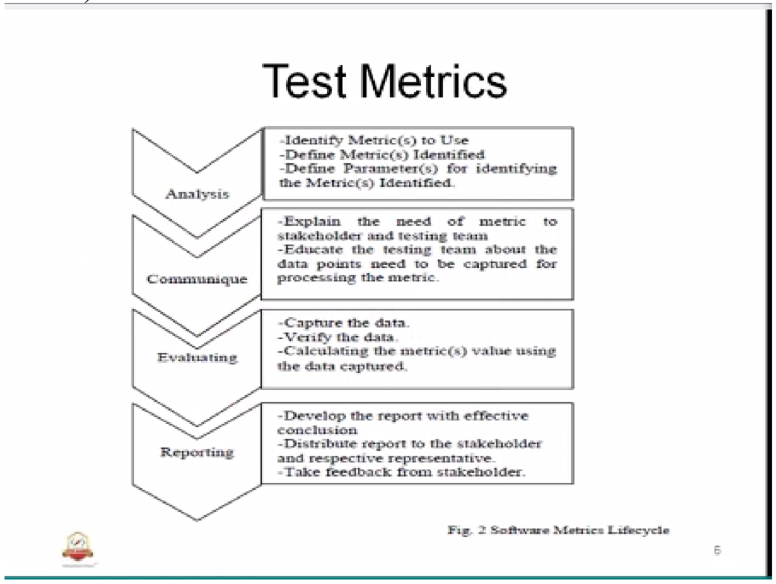
- **Objective**
- This process defines the Metrics & Analysis process for projects, support services, LOBs and Organization. This process also defines Process performance models for Statistical management of processes. Objectives of this process are:
- Establish Organization's Metrics Objectives aligned with the business objectives, and deploy them to LOB, projects and support functions level Specifying the Metrics, data collection and storage mechanisms
- Conduct statistical analysis of data at sub-process, project, support functions, LOB and Organizational levels
- Establish and maintain a quantitative understanding of the performance of the organizational processes and to provide the process performance data, baselines, and models
- Establish feedback mechanisms and provide reports to relevant stakeholders
- Provide data for process improvements

IJCSE, ISSN : 0975-3397

5

e

(Refer Slide Time: 14:53)



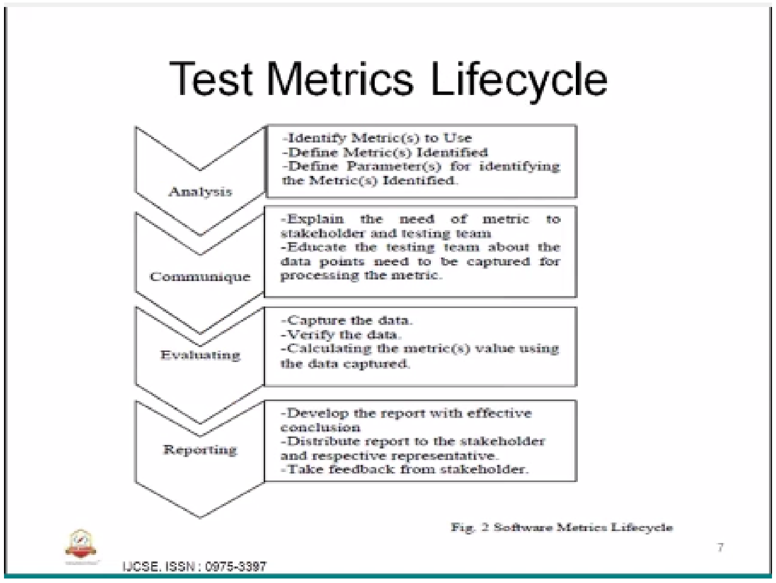
Test Metrics

Analysis
-Identify Metric(s) to Use
-Define Metric(s) Identified
-Define Parameter(s) for identifying the Metric(s) Identified.
Communique
-Explain the need of metric to stakeholder and testing team
-Educate the testing team about the data points need to be captured for processing the metric.
Evaluating
-Capture the data.
-Verify the data.
-Calculating the metric(s) value using the data captured.
Reporting
-Develop the report with effective conclusion
-Distribute report to the stakeholder and respective representative.
-Take feedback from stakeholder.

Fig. 2 Software Metrics Lifecycle

6

(Refer Slide Time: 14:54)



Test Metrics Lifecycle

-Identify Metric(s) to Use
-Define Metric(s) Identified
-Define Parameter(s) for identifying the Metric(s) Identified.

-Explain the need of metric to stakeholder and testing team
-Educate the testing team about the data points need to be captured for processing the metric.

-Capture the data.
-Verify the data.
-Calculating the metric(s) value using the data captured.

-Develop the report with effective conclusion
-Distribute report to the stakeholder and respective representative.
-Take feedback from stakeholder.

Fig. 2 Software Metrics Lifecycle

7

IJCSE. ISSN : 0975-3397

Okay, the next one is being test metric life cycle, so test metric is a life cycle it again depends on the particular project so this is been defined in one of the ISN general, I put the reference in the bottom, so what it says is, there are four test metric life cycle aspects that will be delivered and all that will be used so analysis, communiqué, evaluating, and reporting.

Okay, test metric life cycle analysis has identified operates what is going to be used, define the metric identified, define the parameterize for identifying the metrics which are identified, so these are basically first phase of the test metric life cycle, that is analysis life cycle, analyzed phase. What we do first is, identify the metric to use, so what are the metric we used, we are going to have just identify those, it could be a fast fail count, it could be total number of test cases mapping to the requirements, it could be a defects fixed category like high category, low category or defects of criticality or defects of inner issues likewise, so those need to be identify process what I am going to accept.

Then once I identify that metric then I need to define what that metric is going to have, suppose last fail count, what is the last fail count? So you need to have definition of cross ways when there test is being passed, when it is verified sudden expected results and actual results in the aspect position are matching and it is define as pass. And those pass metric fixed or identified and similarly it fail metric also can be identified and we need to define them.

The next one being define parameters for identifying the metric identifier that means the various parameters that are used for identifying in the terms of inputs are there transition criteria, that is been forward. The next phase of the life cycle test metric life cycle is the communiqué; here what we do is explain the need of metric to stakeholder and testing team. Basically who are going to develop this test metric? Who are going to deliver the test metric? Who are going to use it?

Basically testing team and the relevance stakeholders surrounding the testing team.,so those need to be clearly explained about what we have done in the analysis, we have collated all the defects information inputs of the testing program and that needs to be communicates, that needs to be explained to the relevance stakeholders, because they are going to develop it or use it, whatever has been defined, if we equate the testing team about the data falls need to be captured or processing the metric. So we have defined it, we have identified it, how we are going to do it? So all this how we need to be clearly or told to the testing team, that is very important, so this is what we do in the communiqué life cycle part of the test metrics.

The next one is the evaluating. So what we have done in the communiqué is, we have told and start the activity of a metric collection the metric usage and all that, so once we have the metric available that needs to be verified, that needs to be reported appropriately I mean report it in the next life cycle part, for reporting before that we need to have appropriate calculation in terms of percentage calculation, it could be trend updates any formula is to be used all this have to be done.

Because usually this will be done at using automated reporting, automated way, so how they do usually is in excel sheet, so excel sheet is a very good tool to enter the terms and reporting the artifacts, so all these will be the part of the capture verify and calculate the metric, so verify also is one of the important aspect, why because this is been captured rightly and correctly without any errors, so that is also important aspects.

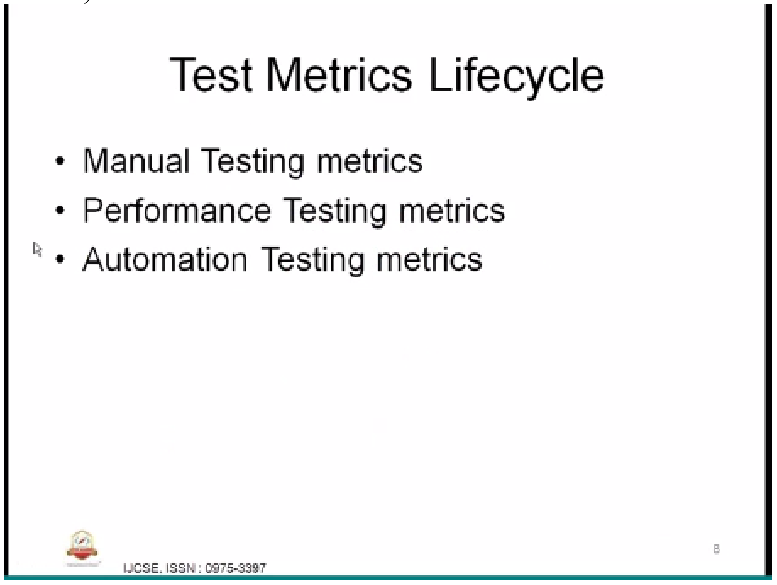
Basically what we do is, we will evaluate the captured data and verify against data is been correctly captured and we do a calculations which will be, which will help basically reporting the metric, usually we use a different tools for capture and the calculation, The final life cycle that is, the reporting. So what we do in reporting is to develop the report with effective conclusion, that means we need to present it properly so that the customer could be higher managers or it could be any program management team, then we need to be knowing what is going to happen, what is going on? And what is the defect? How is it been captured? How they have been evaluated? All these aspects have to be presented appropriately.

So we need to develop a very good report in terms of objectives, scope, description then we should have a evaluating mechanism, what are the issues that are found to be testing and how they have been reported all these, also we should end the report with the conclusion and somebody saying that this product is fit for use or this product has some issues, why those issues are there, if those issues are fixed or justified brought good to use, the product has major issues or the minor issues, all them can be rightly reported, that’s what this mean.

Then it needs to be, once the report has been done, we need to distribute to the stakeholders and respective representative, we can also have a distributed to developing team representing team, who can take and look in to this and try to improve or fixed the issues that particular aspects, also we need to, it is very important whatever the report we have been developed and reported, is rightly understood and rightly been reached to the stakeholders, so how do we do?

How we are going to do is by taking inputs the feedback from the leveled stakeholders, so that is what we do with the report testing life cycle. Okay, so software testing metrics types, what are the string model types that we have today for an embedded system project, we know that, we are going to have different types of testing, so they all broadly categorized into automated manual, white box, black box all this static analysis to be reviewed artifacts, all this will be the part of one of this three. What are those? A manual testing metric performance testing metric,

automation testing metric. So we will study each one, what is manual? What is performance? And what is automation?
So we know that testing can done manually like analysis, some of the assembly level debug or testing cannot be possible to do, the dynamic testing automate evaluate,
(Refer Slide Time: 24:08)

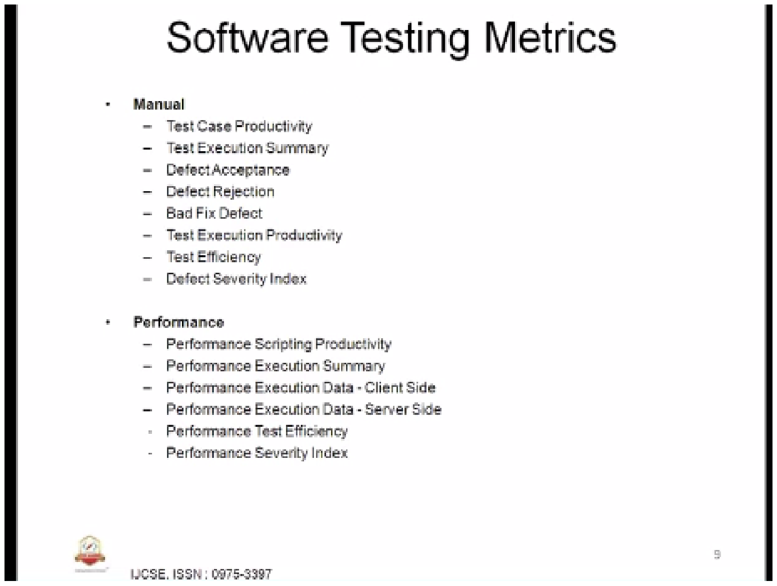


Test Metrics Lifecycle

- Manual Testing metrics
- Performance Testing metrics
- Automation Testing metrics

IJCSE, ISSN: 0975-3397

So we need to use environment such that the environment is given and verified manually. So those manual testing metric also need to be captured, also we know that the static analysis is also a part of the manual testing. Where we do the testing without executing the program so that will be the part of testing, performance testing, performance of the exhibition summary , exhibition data, it could be an a client side, server side, and the efficiency all these are like driving performance and memory all this will be separately reported in the performance testing metrics.
The automation testing metric, so what we do is, we will capture and report the metric for the automation, and how much is done? What are the artifacts? What will they lose in the automation of the tests?
(Refer Slide Time: 25:20)

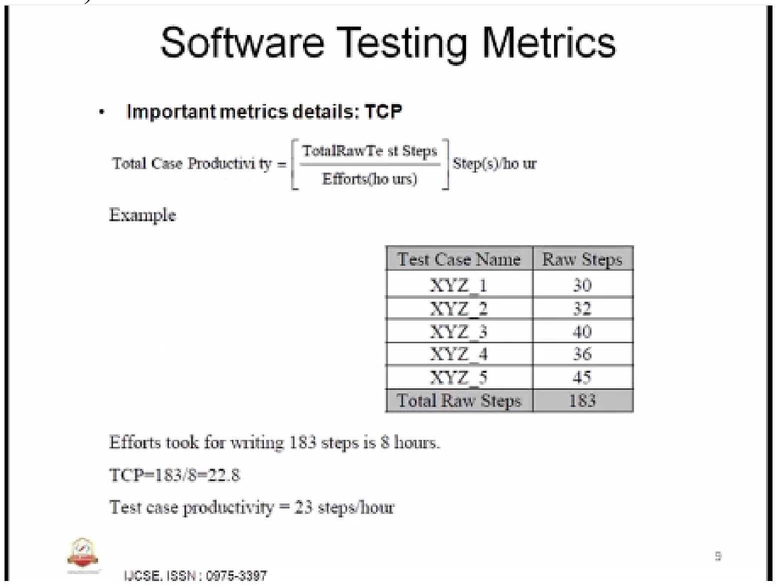

Software Testing Metrics

- **Manual**
  - Test Case Productivity
  - Test Execution Summary
  - Defect Acceptance
  - Defect Rejection
  - Bad Fix Defect
  - Test Execution Productivity
  - Test Efficiency
  - Defect Severity Index
- **Performance**
  - Performance Scripting Productivity
  - Performance Execution Summary
  - Performance Execution Data - Client Side
  - Performance Execution Data - Server Side
  - Performance Test Efficiency
  - Performance Severity Index

IJCSE, ISSN : 0975-3397

9

(Refer Slide Time: 25:21)


Software Testing Metrics

- **Important metrics details: TCP**

$$\text{Total Case Productivi ty} = \left[\frac{\text{TotalRawTe st Steps}}{\text{Efforts(ho urs)}}\right] \text{Step(s)/ho ur}$$

Example

Test Case Name	Raw Steps
XYZ_1	30
XYZ_2	32
XYZ_3	40
XYZ_4	36
XYZ_5	45
Total Raw Steps	183

Efforts took for writing 183 steps is 8 hours.

TCP=183/8=22.8

Test case productivity = 23 steps/hour

IJCSE, ISSN : 0975-3397

9

This type of study, software testing metrics detail okay manual testing metric will have a productivity , test case productivity, test exhibition summary, defect acceptance, defect rejection, bad fix defect, test execution productivity, test efficiency, defect severity index. So these are some of the testing metric, so some of these important metric in this model explain you in the next slide, so basically what it does, when we go throw the testing life cycle, here we do a test case development, test procedure development, and test script development and test execution, test exhibition, regression testing.

All these so we will defiantly will have different types of artifacts accumulated in the terms of test cases itself and the program underneath which is been tested also can be used. So test case productivity basically how much productive that the tested team has? In the terms of developing test cases, and test exhibition somebody, that means in the terms of exhibition how it went well in the terms of how much time it took for a manual testing vision, so whatever result you get always be part of this summary, so that also be metric side, then defect acceptance whether the defects 100 defects reported, how many of them are really defects? How many of them are rejected defects?

That also need to be captured, that fix defect and the program has a issues and the issues have been fixed, but the fixed is not correct, then the test execution productivity, how much productivity the test team or the exhibition, it could be a resource or it could be a resource strong human whatever it is, so some test may take few hours and some test take few minutes, depends on the complexity of the type of design that have been done, any test efficiency, how he test efficiency is, the manual testing, then the defects where it is and acceptance , and how previous defect are, coming to next one performance testing metric, you have a performance scripting productivity, we have performance exhibition summary, performance exhibition data, in the terms of both side and server side, we can assume this as a target. Likewise you can apply use this example that is been taken, then the testing efficiency of the performance testing.

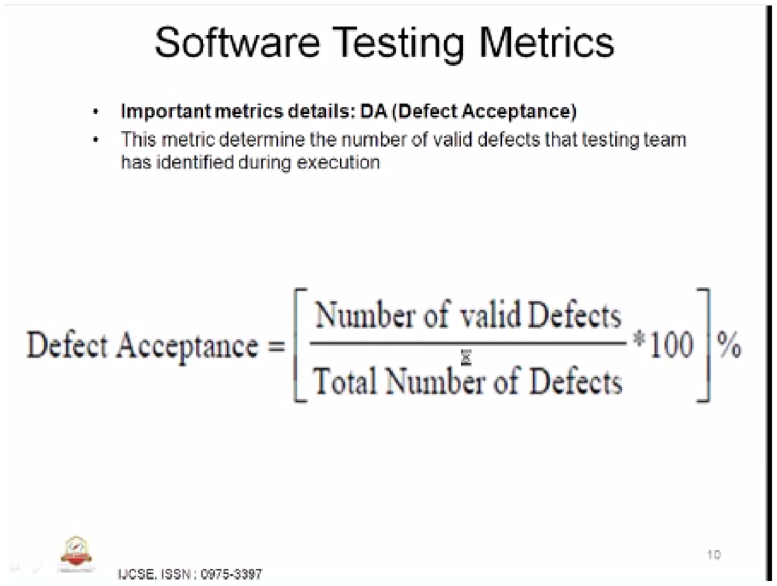
Then the performance severity index, all the performance type of testing will have to be captured separately using the testing metric guidelines. So that is what the software testing metric types. We will pick up couple of good examples of this metric, test case productivity. So what is the test case productivity or total case productivity? So formula is test case or TCP, it is called as productivity but total number of test steps divide by efforts into step hours, or this steps or that we will use total number of sorry, the total number of productivity of the test case development, that means there are number of steps, say 100, and effort took for that is few hours that will be divided again that and that will be nothing but the step hour.

Steps per hour that is what the test case productivity, So example it is given here, so test cases explains 12345, so each one has its own number of steps 30,32,40,36,45, the total number of steps are 183. So for example effort that is taken for developing these steps in eight hours, so what is going to happen is, the test case productivity is going to be 183 divided by 8, what it is going to be taken here in this formula and which is going to be 22.

That means the test case productivity 22 rounded of 23 steps per hours, 23 steps per hour is productivity. So what way this is going to be useful in what way it will be getting used, so you need to understand that, so there are different aspects that are important, why that is important? Why this test case productivity used , productivity is important of estimation and confidence, these three aspects are basically the driving factors for identifying the this test productivity, so if the test case productivity is good in a next cycle suppose on the previous sector that means we are improved, and if I take a new project or testing project has come it's so complexity and testing required, we know we have taken this much to develop the test and we will apply or estimating that new project.

That is a factor we will use it and again it depends on the complexity subjectively it has to take care based on this test steps all that they have written, whether it is similarity or it is different then what we done in the earlier project likewise, so this test metric is very important. And the last one is confident, basically if you use a confidence are a program management and the test management by seeing the productivity how much productive the testing team in the terms of developing test case, so that is what the important of a test case productivity. The next type of,

(Refer Slide Time: 32:26)



Software Testing Metrics

- **Important metrics details: DA (Defect Acceptance)**
- This metric determine the number of valid defects that testing team has identified during execution

$$\text{Defect Acceptance} = \left[\frac{\text{Number of valid Defects}}{\text{Total Number of Defects}} * 100\right]\%$$

IJCSE, ISSN : 0975-3397

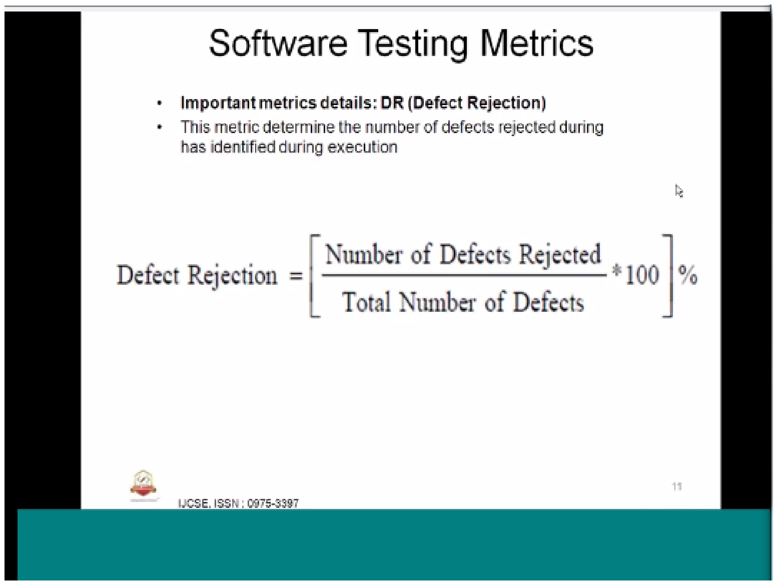
10

Metrics is the defective, the defective acceptance. So we know that the tester test and report the defects, whether those defect are valid or those defects are invalid, that is also need to be identified from the developers or the tester, so they are written a test steps here executed the test steps and it has failed, the failure could be here into s, the trailer could be there in the software.

Their field could be in the requirement, or files could be in the design, whatever it is all are that side, also it is important to know whether the failures are real failures, acceptable failures, or there is a issue with the invalid test case design or test steps. For example suppose we have, we know in the test case design we have a accepted results column and we have actual design and expected result is spoken through the actual result has become false, so we report this as a failure, suppose instead of mentioning it true , the test arriver mentioned wrongly that false, what will happen?

The actual report, the actual result and excepted result will match, and the report has non failure, so what will happen is, this falsely reported as a power but there is a issue, so we need to correct that as a non excepted failure, so this is very important in the terms of defect acceptance. So defect acceptance formula is like this, total number of defects we have and how much of them are really valid, for example we have, 40 valid defects against the total number of 50, 100, becomes 80%. Right, so that is the percentage we are going to wrong, so this will give me 80% of defect that has been reported or valid and 20% is a issue. 20% are invalid defects, that the defects are not really genuine or there is an issue with a test case or reporting whatever it could be or documentation so it is very important to understand the defect acceptance as one of the key metrics let us to be reported.

(Refer Slide Time: 36:09)


Software Testing Metrics

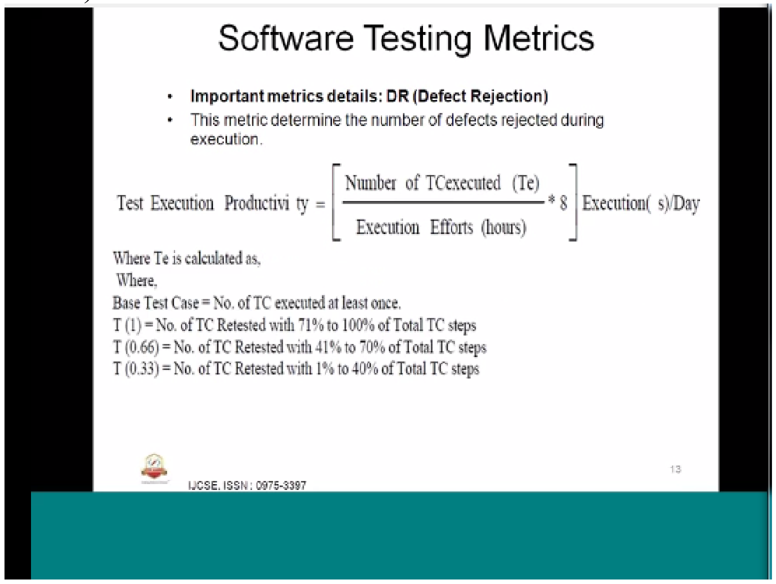
- **Important metrics details: DR (Defect Rejection)**
- This metric determine the number of defects rejected during has identified during execution

$$\text{Defect Rejection} = \left[\frac{\text{Number of Defects Rejected}}{\text{Total Number of Defects}} * 100\right]\%$$

IJCSE, ISSN : 0975-3397

11

The next one defect rejection these also one of the important test metrics, this will determine the number of defects rejecte4d during the execution. That means the reported defects are not justifiable or the issues that is found out in the requirement, design or not accepted by the development plane, those acceptances to rejection ratio of the rejection percentage, or DR percentage. So defect rejection is nothing but number of defects rejected, here total number of defects in to 100. That is the percentage. So, here it is a valid defect it is defects rejected. So that is what the percentage.
(Refer Slide Time: 37:04)


Software Testing Metrics

- **Important metrics details: DR (Defect Rejection)**
- This metric determine the number of defects rejected during execution.

$$\text{Test Execution Productivi ty} = \left[\frac{\text{Number of TCexecuted (Te)}}{\text{Execution Efforts (hours)}} * 8\right]\text{Execution(s)/Day}$$

Where Te is calculated as,
Where,
Base Test Case = No. of TC executed at least once.
T (1) = No. of TC Retested with 71% to 100% of Total TC steps
T (0.66) = No. of TC Retested with 41% to 70% of Total TC steps
T (0.33) = No. of TC Retested with 1% to 40% of Total TC steps

IJCSE, ISSN : 0975-3397

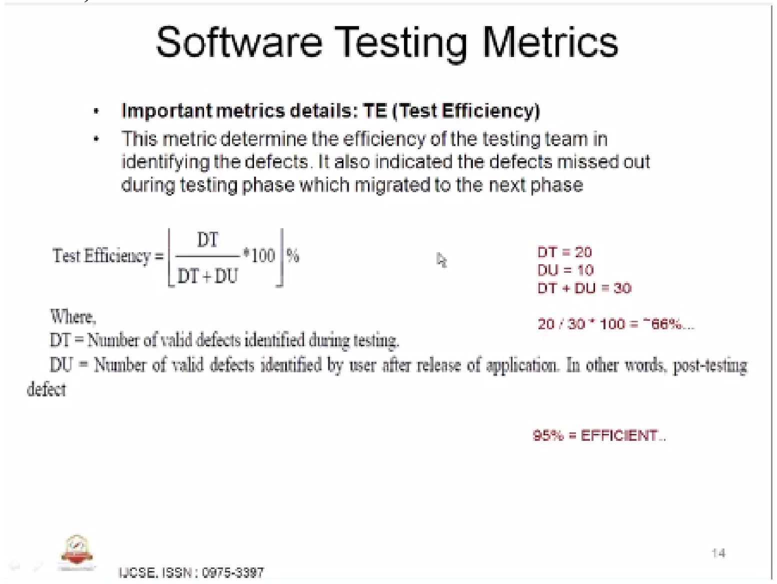
13

Next one is test execution productivity, this also very important testing metrics. Because this will give you good estimation of how much it is going to give? How much time it will go to take? For a given product having so and so steps, and having so and so test cases. So basically this test metric use the test cases aggregation productivity which is further analysis can give conclusion result that means we have it test case productivity in terms of hours basically or days it could be so there are formulas here.

Test case test execute the productivity are driven, total number of test case are executed here by hours the total hours days into eight, so eight is what typically the allocated time in a working hours of the day basically, so that got the optimal value they use, so where (Te) so where t is humored, t is calculated as test cases, so the number of test cases execute at least once, the number of test t of one is nothing but t of one is said to be number of test cases that we tested within, with 71 percent to 100 percent of total test case steps, t0.66 is equal to number of test case retested with 41 percent of 41 to 70 percent of total test case steps 0.33 will give you a metric as number of test case retested with one percent to 40 percent of total T systems.

Okay, so basically here idea here is, we should have executed the test cases at least once, it could take multiple interactions that are 5, but typically the productivity is based on the one time executions, so that in regression we do not want to put the feet, the executions as we did in the primary stage. So that is what it means, so retesting aspects will be defined like this T of one T of point 0.66, T of point 0.33, that is the percentage that is 33%, 66%, likewise about 1 to 40 percent of total test case time will take retesting, and 40 to 70 percent of total test cases will takes 66.6 as the test execution productivity. And ideally 71 to 100 percent of the total test cases will take the productivity as one, so that is what says, how productive we have in the terms of execution against, this is what we have a testing equations productivity.

(Refer Slide Time: 40:45)



Software Testing Metrics

- **Important metrics details: TE (Test Efficiency)**
- This metric determine the efficiency of the testing team in identifying the defects. It also indicated the defects missed out during testing phase which migrated to the next phase

$$\text{Test Efficiency} = \left[\frac{DT}{DT + DU} * 100\right] \%$$

Where,
DT = Number of valid defects identified during testing.
DU = Number of valid defects identified by user after release of application. In other words, post-testing defect

DT = 20
DU = 10
DT + DU = 30

20 / 30 * 100 = ~66%...

95% = EFFICIENT..

14

IJCSE, ISSN : 0975-3397

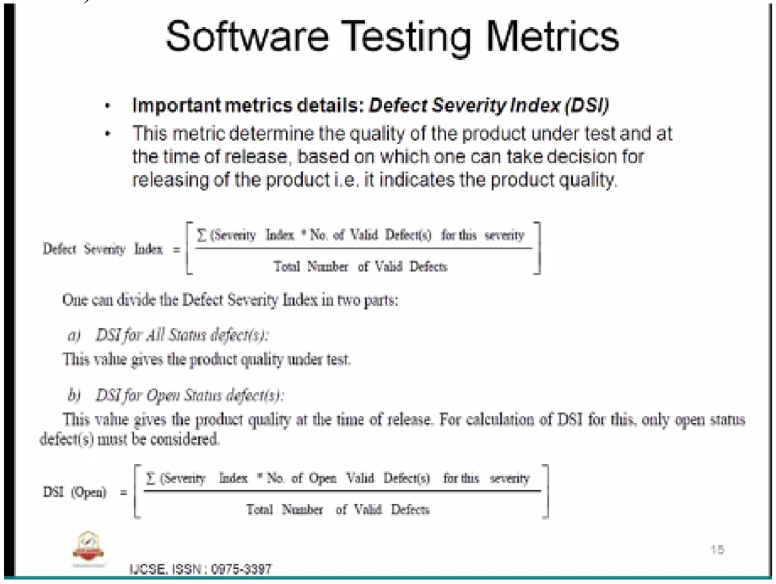
Test efficiency, how efficient we are in the terms of executing the test and reporting the defects. This metric define determine the efficiency on the efficiency of the testing team and identifying the defects. It also indicates the defects missed out during testing phase which migrated to the next phase, so test efficiency is nothing but DT divided by DT plus DU into 100, this gives you a percentage vale, where DT is nothing but it is the number of valid defect identified during the testing, DU is nothing but number of valid defects identified by user after release on application.

In the other words, post-testing defect, during the testing we have number of valid defects and the defects that are identified and still after these are the application, it is nothing but the post fully defects. Suppose we have detected some few errors two defects in the pre-released base of the operate testing and that is been fixed retested and it is passed all these as close and we have released still there are issues which is UN covered during the test execution.

So that uncovered position of that is nothing but thee, it is the efficiency of the tester and efficiency of the testing life cycle it shows. So basically the number of valid defect if it is DT equal to, suppose 20 defects have been identifier and they are fixed they are released and the number of defects found out during the post-testing field on the user side, if we say suppose 10, so what will happen is we will keep it as DU equals to 30, and we have DT by DT plus DU, if it is 20 divided by 30 into 100.

So something like 60 percent whatever it is. Right, That means it is two third of the or the 60 percent of the defects efficiency we have, that means we are efficient 60% in terms of testing the product, still we have a gap of 40%, it is not 66 approximately, so still we have a 66% issue with the testing, that is we are not efficient, so in the embedded industry typically the aspect is 95%, this is what the metrics they follow, because nowadays it has to be very stringent and it companies are not afford to reproduce, re-execute again the retest and all that, because it is going to add cost a lot, so we cannot afford to have a any bugs are post-release issues. We have to be 90 above percentage in the terms of efficiency that is one of the important metrics. The next one is a,

(Refer Slide Time: 44:52)



Software Testing Metrics

- **Important metrics details:** ***Defect Severity Index (DSI)***
- This metric determine the quality of the product under test and at the time of release, based on which one can take decision for releasing of the product i.e. it indicates the product quality.

$$\text{Defect Severity Index} = \left[\frac{\Sigma\text{ (Severity Index * No. of Valid Defect(s) for this severity}}{\text{Total Number of Valid Defects}}\right]$$

One can divide the Defect Severity Index in two parts:

a) *DSI for All Status defect(s):*

This value gives the product quality under test.

b) *DSI for Open Status defect(s):*

This value gives the product quality at the time of release. For calculation of DSI for this, only open status defect(s) must be considered.

$$\text{DSI (Open)} = \left[\frac{\Sigma\text{ (Severity Index * No. of Open Valid Defect(s) for this severity}}{\text{Total Number of Valid Defects}}\right]$$

15

IJCSE, ISSN : 0975-3397

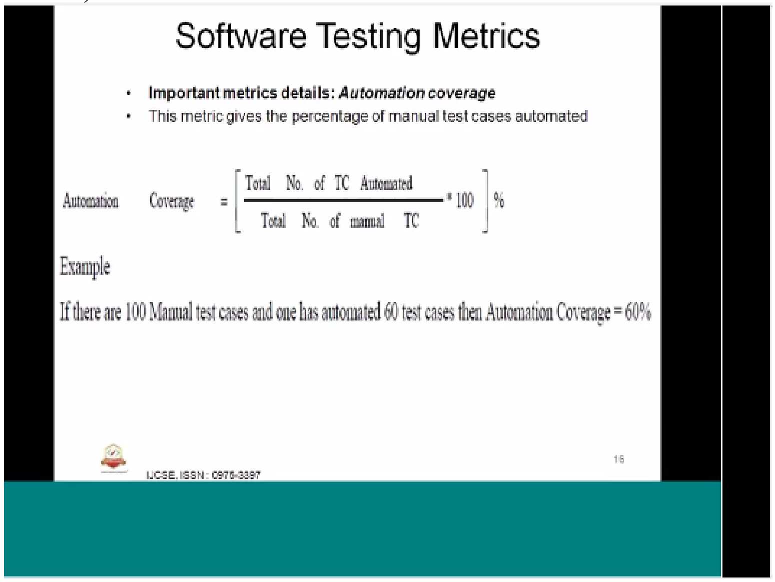
Defects severity index, I will tell you overall so you do not want to bother much about the formula and all, this is basically

The severity of the defects that means you has I hope 10 defects. So all defects may not be very sever. I could be major issue or minor issue or trivial issue might be having a category. Severity in terms of how much it is going to cause in terms of damage to the product. It is going to have a impact on the product function on the behavior so it could be a major or minor so each one will have an it is own weight age.

Basically that weight age is what we will give you the index the defect severity index. DSI is one the important software testing metrics. That is where in the industry metrics. And for formula is like defect severity index is nothing but it is some of the index number of all that particular severity. That the total number of matrix. One can divide the defect severity index in to two parts. DSI for all status defects, that means all the status have been reported this value give the product quality and test.

And next one is the DSI for open status defects. This value gives the product quality at the time of release that means when you are going to deliver the product at that time we are going to have the open status defects. Those severity, will be reported, for calculation if DSI which open status of defects. So, only open status defects must be considered. So the formula is for that DSI is some of DSI index of open number of defects for this severity. Here the total number of valid defects. And that is what a DSI open severity index will indicate.

(Refer Slide Time: 47:54)



Software Testing Metrics

- **Important metrics details: *Automation coverage***
- This metric gives the percentage of manual test cases automated

$$\text{Automation Coverage} = \left[\frac{\text{Total No. of TC Automated}}{\text{Total No. of manual TC}} * 100\right]\%$$

Example

If there are 100 Manual test cases and one has automated 60 test cases then Automation Coverage = 60%

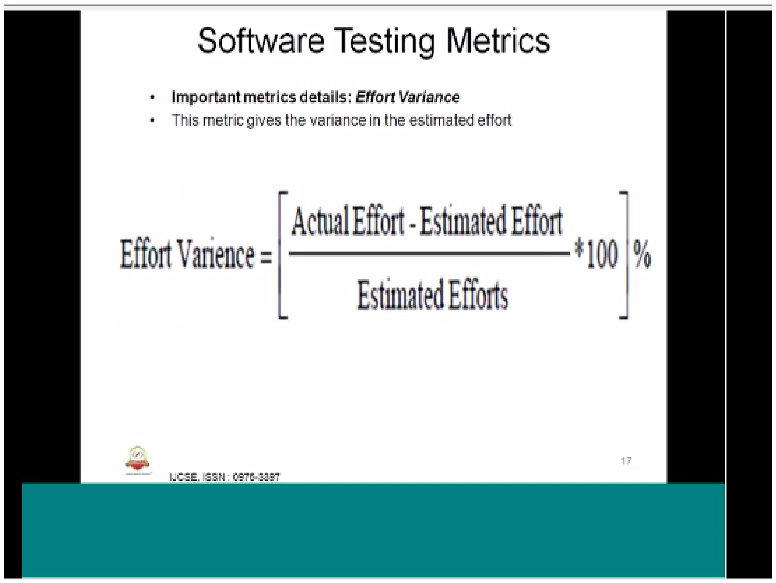
IJCSE, ISSN : 0975-3397

16

The next one is Automation coverage. So this automation coverage is also one of the important metrics we have. So this metrics gives the percentage of manual test cases which are automated. This we know things cannot to be done 100% automation. So we need to do a subjective automation and subjective manual testing. So that is the mix of both. So how much percentage we can add to automate. Because automation we do not have a human depends. It is done by the machine.

And we know the productivity and coverage. So coverage is basically total number of test cases automated given the total number of space into 100. Give you the automatic coverage. Example if there is 100 manual test cases, and one has automated 60 test cases then the coverage is 60% that is what the meaning of automation coverage.

(Refer Slide Time: 48: 55)



Software Testing Metrics

- **Important metrics details: *Effort Variance***
- This metric gives the variance in the estimated effort

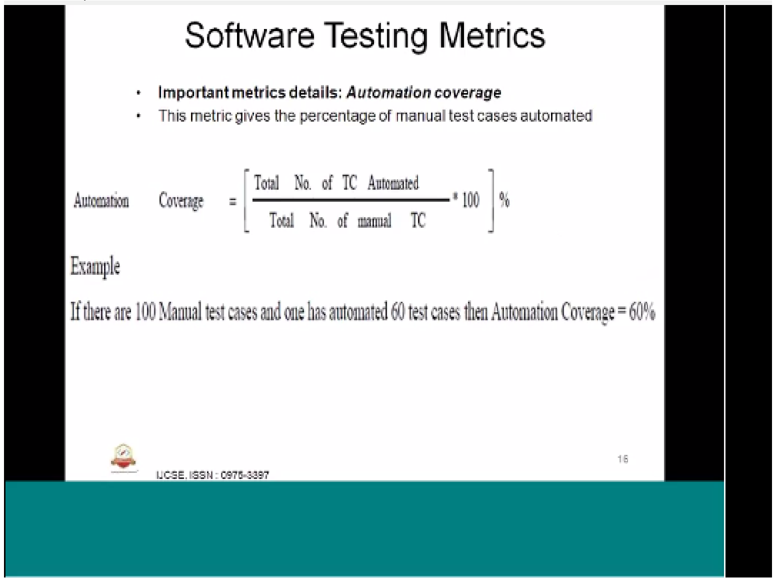
$$\text{Effort Varience} = \left[\frac{\text{Actual Effort - Estimated Effort}}{\text{Estimated Efforts}} * 100\right]\%$$

IJCSE, ISSN : 0975-3397

17

The next one is effort variance. This is one of the important matrices that is usually managers they do.

(Refer Slide Time: 49:19)



Software Testing Metrics

- **Important metrics details: *Automation coverage***
- This metric gives the percentage of manual test cases automated

$$\text{Automation Coverage} = \left[\frac{\text{Total No. of TC Automated}}{\text{Total No. of manual TC}} * 100\right]\%$$

Example

If there are 100 Manual test cases and one has automated 60 test cases then Automation Coverage = 60%

IJCSE, ISSN : 0975-3397

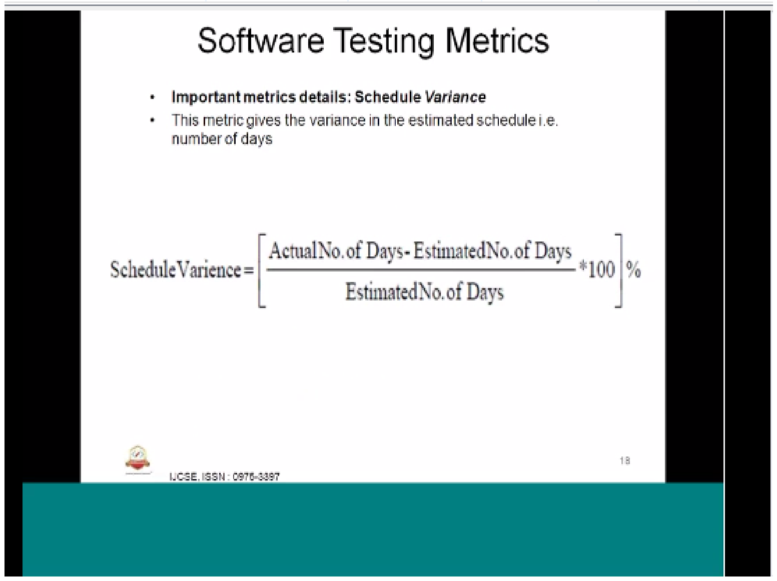
16

Automation coverage we know that how much test cases have been arithmetic against the manual test cases the next type of the effort variance. It is one of the important metrics that is been used by the program management of the higher of basically so, effort variance is the actual effort minus estimated efforts and multiplied by 100. That is gives the percentage. So basically what we do is we know how much time how much effort it took for a multiple testing. It could be 200 hours. This is what is estimated basically that means start of the program you know that 200 hours we going to take for a suppose so much of test cases and actual that will happened is something. So what is the variation how much it is getting a variance it is nothing but actual effort minus estimated efforts it will give you minus 20. Usually it will be more depending on the projects.

Appropriately it will be done. So you had the 100. Sorry you had by 200. It is giving you 10%. Basically minus 10% so, similarly if we have taken the actual value as 220 hours that case how much it will be come? 220-estimated is 200 it will become 20/200 that is the 10% so effort variance. So whatever the variance from the estimated value of the 10% the typical industry the variance accepted it minus 10 to plus 10. You may ask question why minus 10 plus 10? Sometimes what will happen is it will be overestimated because of the lack of knowledge of something.

So but we do not take that much of effort. To be on the safer side what they will do it. they will add some 5 to 10% of over estimation testing. So what will happen is over a period that will over here and there. So maximum allot this only minus 10 to percentage it is what they various that we can offered. That is what the effort variance is above.

(Refer Slide Time: 52:28)



Software Testing Metrics

- **Important metrics details: Schedule *Variance***
- This metric gives the variance in the estimated schedule i.e. number of days

$$\text{ScheduleVarience} = \left[\frac{\text{ActualNo. of Days} - \text{EstimatedNo. of Days}}{\text{EstimatedNo. of Days}} * 100\right]\%$$

IJCSE, ISSN : 0975-3397

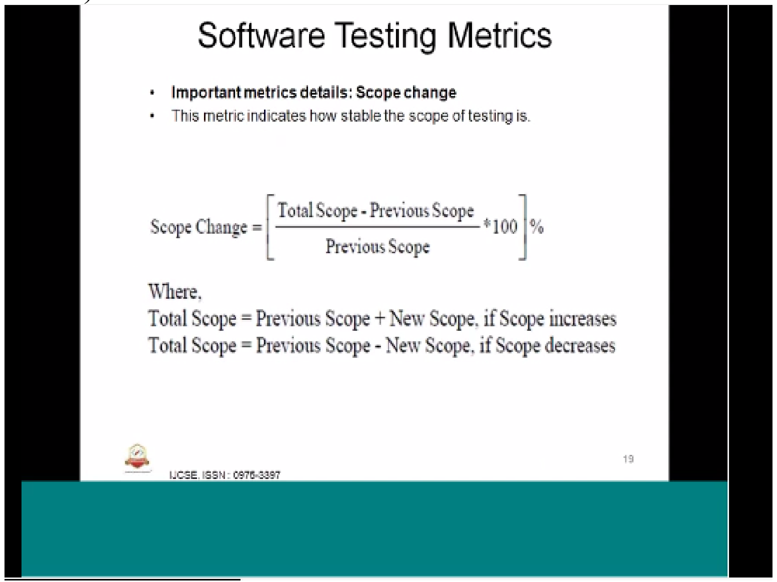
18

The next step of metrics is schedule variance so what you mean by schedule variance. We know here also estimation is there. So estimation is done into this typical estimation I am taking about there are different types. In terms of efforts, in terms of duration both are very important. In terms of schedule this is all schedule aspects okay, so efforts variance will talk about efforts and duration variance.

We will talk about schedule variance. schedule variance is nothing but the actual number of the minus estimate number of base in terms of percentage suppose or 10 test pr 20 test or the estimate days is nothing but 20 days. And actually they took 22 days. So what is the schedule variance? Which is also called as we assume actual number of days 25? We have to neglect that with the estimate is 20 dived by estimated 20 how much it will come 25-20 is 5 dived by 20 nothing but ¼ or point 5. Right, this is 25% so, have a variance of 25% usually typical it is not have too much of variance schedule variance. So but it have a limited or controlled schedule variance so that is why it suggested to have a SV and EV calculated by regularly.

So that we know how much we are progressing. So these are very important aspects of the embedded software testing. So usually 5% is what schedule variance also it depends everyone on the complexity of the product etc. so 5% is what they allowed? During the point is very important testing metrics. By using the process throughout the project in been as well as the beginning end all the faces. So that is what SV and EV effort variance and schedule variance. Repeated effort variance is the variance of the estimated efforts. Schedule variance is the variance of the estimated.

(Refer Slide Time: 55:29)



Software Testing Metrics

- **Important metrics details: Scope change**
- This metric indicates how stable the scope of testing is.

$$\text{Scope Change} = \left[\frac{\text{Total Scope} - \text{Previous Scope}}{\text{Previous Scope}} * 100\right]\%$$

Where,
Total Scope = Previous Scope + New Scope, if Scope increases
Total Scope = Previous Scope - New Scope, if Scope decreases

IJCSE, ISSN : 0976-3397

19

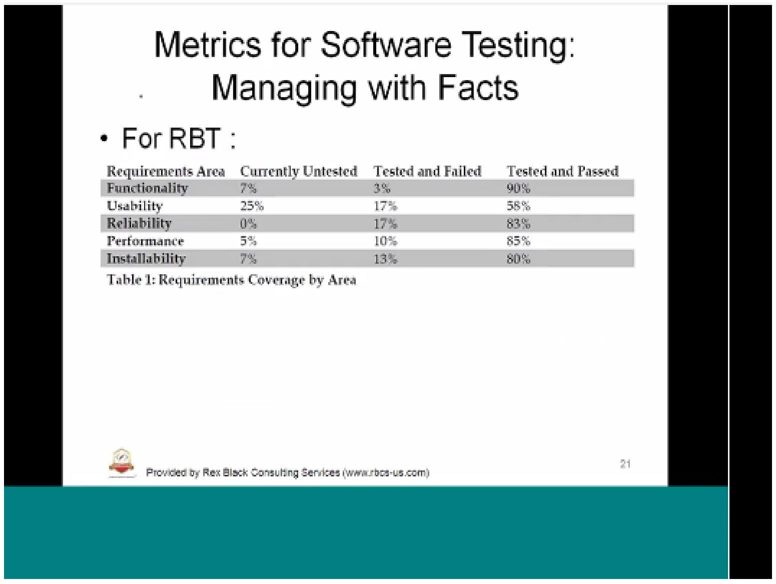
The last on is scope change. This is the important metrics also. The metrics this one indicate how stable this scope of testing. We know what is our scope? So vary to we knowing how much it has to be changed from the original scope that is been defined the scope is increase or decreases during the test pr after the testing whatever it is. So, how it is calculated is it is scope change. Basically that is the metrics we are going to produce. Scope change is nothing but hope minus hope divided by previous hope into 100 in terms of percentage.

Where total scope is scope+New scope this scope increase total scope-New scope decreases. For example, scope change, so pervious scope change suppose can be done this scope. 10 scope means I have 10 requirements to cover. And that total scope is 20. So scope change is to be calculated. We have total scope minus previous scope 10/previous scope in terms of percentage. How much has been written now. 10-10/10. Equal to 1. It is 100% alright. Yes, 1 into 100. So, 100% of this scope will change.

That means trouble something like that. So scope is increased right, it is previous scope was scan and total scope at the end of the testing or it has to be changed it has increased 100%. It is scope is decreased then it will be decrements scope calculation. Previous scope minus previous scope it is scope decreases scope. This originally I start with 100 requirements. That is the scope to cover in the testing.

And the testing let us change scope of the testing. So, the change it could be added new scope requirements or added new test cases or decremented test cases or scope the scope decreases so the scope decreases or the scope increases. So that is what the scope change this one of the important test metrics. They use across. Oaky that is what the software testing metrics so in the next session.

(Refer Slide Time: 58:42)



Metrics for Software Testing: Managing with Facts

- For RBT :

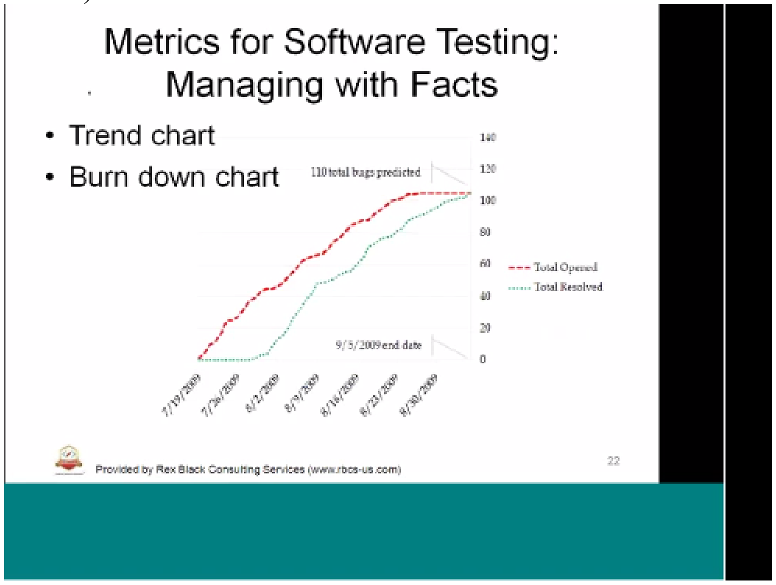
Requirements Area	Currently Untested	Tested and Failed	Tested and Passed
Functionality	7%	3%	90%
Usability	25%	17%	58%
Reliability	0%	17%	83%
Performance	5%	10%	85%
Installability	7%	13%	80%

Table 1: Requirements Coverage by Area

Provided by Rex Black Consulting Services (www.rbcs-us.com) 21

We will conclude the automation test metrics.

(Refer Slide Time: 58:43)



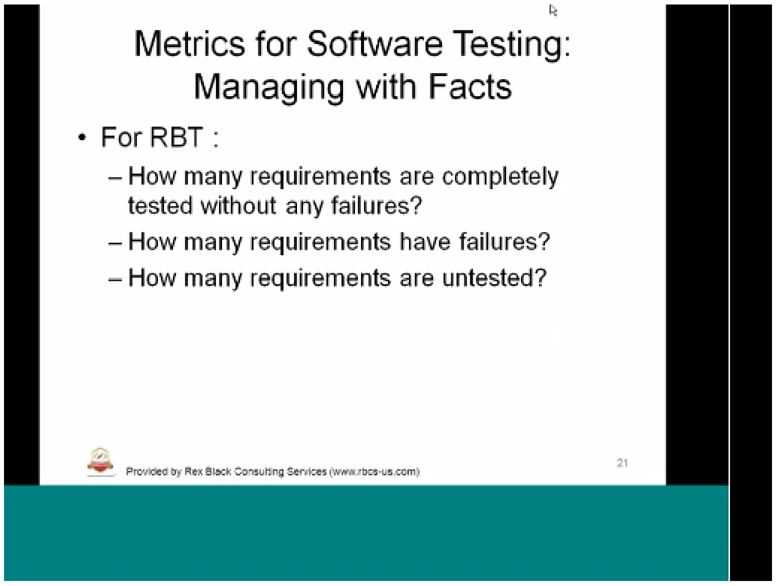
Metrics for Software Testing: Managing with Facts

- Trend chart
- Burn down chart

Provided by Rex Black Consulting Services (www.rbcs-us.com) 22

And now we are going to manage how we are going to reported what are the trends charts, burn down charts etc.

(Refer Slide Test: 58:56)



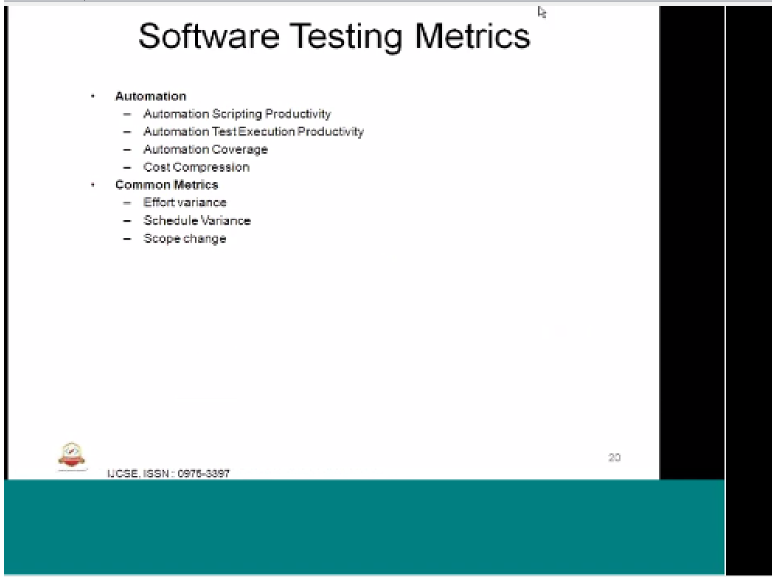
Metrics for Software Testing: Managing with Facts

- For RBT :
  - How many requirements are completely tested without any failures?
  - How many requirements have failures?
  - How many requirements are untested?

Provided by Rex Black Consulting Services (www.rbcs-us.com)

21

and with that we will complete the unit 3 session in next lecture.
(Refer Slide Time: 58:57)



Software Testing Metrics

- **Automation**
  - Automation Scripting Productivity
  - Automation Test Execution Productivity
  - Automation Coverage
  - Cost Compression
- **Common Metrics**
  - Effort variance
  - Schedule Variance
  - Scope change

IJCSE, ISSN : 0975-3397

20