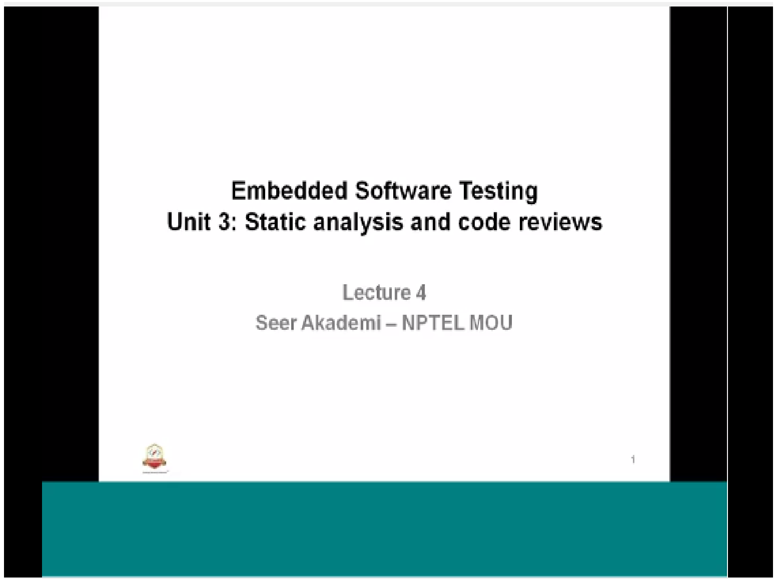
(Refer Slide Time: 00:08)

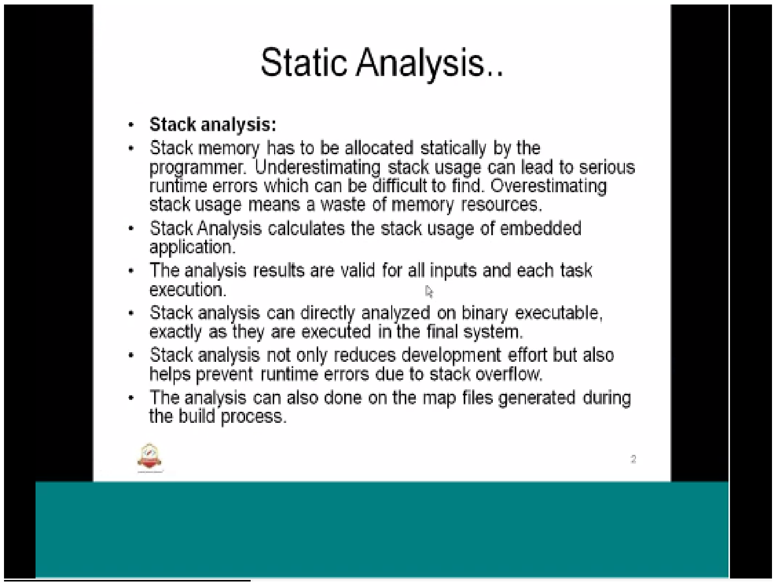


Embedded Software Testing
Unit 3: Static analysis and code reviews

Lecture 4
Seer Akademi – NPTEL MOU

The next session of Embedded Software Testing Unit 3 static analysis and code reviews, and metrics, Today's class is about continuation of the static analysis use and code review is Lecture 4. So before that we will go through a previous session what we have, we quickly so static analysis is one of the important static analysis prospect.

(Refer Slide Time: 00:37)

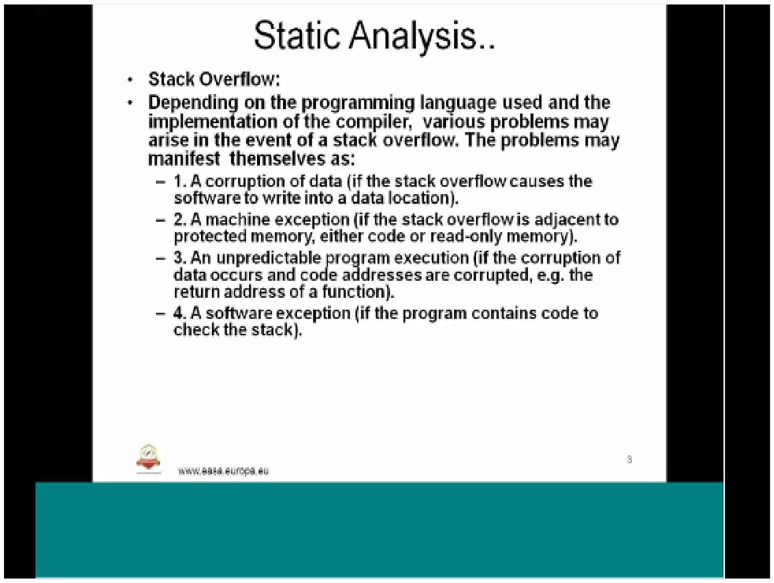


Static Analysis..

- **Stack analysis:**
- Stack memory has to be allocated statically by the programmer. Underestimating stack usage can lead to serious runtime errors which can be difficult to find. Overestimating stack usage means a waste of memory resources.
- Stack Analysis calculates the stack usage of embedded application.
- The analysis results are valid for all inputs and each task execution.
- Stack analysis can directly analyzed on binary executable, exactly as they are executed in the final system.
- Stack analysis not only reduces development effort but also helps prevent runtime errors due to stack overflow.
- The analysis can also done on the map files generated during the build process.

So, stack memory has to be allocated statically by the program. And it cannot be underestimating as well as it cannot overflow so, to measure that we need to the analysis. The measurements can done in differ rent ways. So we need to be binary excitable map file. In the map file we can do line analysis of the in terms of stack. I can be depends on the somewhat we are using. And stack overflow ways one of the important session is stack analysis.

(Refer Slide Time: 01:19)



Static Analysis..

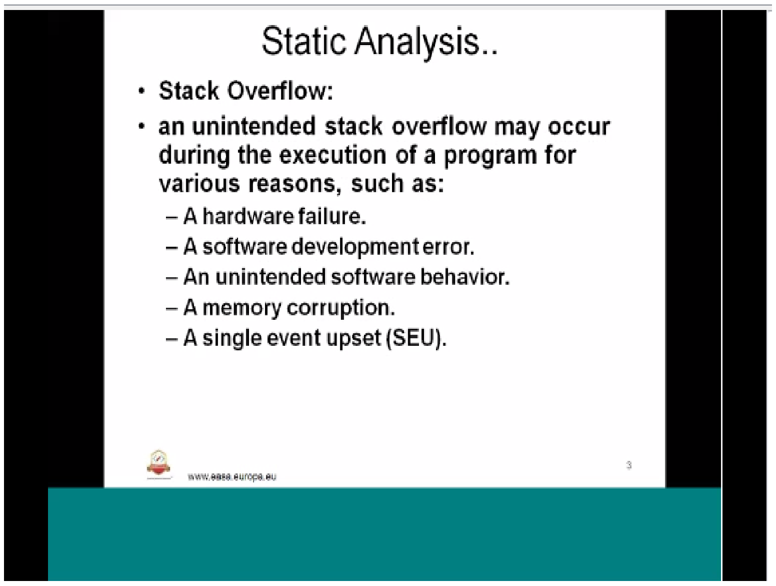
- **Stack Overflow:**
- **Depending on the programming language used and the implementation of the compiler, various problems may arise in the event of a stack overflow. The problems may manifest themselves as:**
  - 1. A corruption of data (if the stack overflow causes the software to write into a data location).
  - 2. A machine exception (if the stack overflow is adjacent to protected memory, either code or read-only memory).
  - 3. An unpredictable program execution (if the corruption of data occurs and code addresses are corrupted, e.g. the return address of a function).
  - 4. A software exception (if the program contains code to check the stack).

www.easa.europa.eu

3

So stack overflow was varying to a correction of a machine exception. And unpredictable program execution is a software exception likewise.

(Refer Slide Time: 01:33)



Static Analysis..

- **Stack Overflow:**
- **an unintended stack overflow may occur during the execution of a program for various reasons, such as:**
  - A hardware failure.
  - A software development error.
  - An unintended software behavior.
  - A memory corruption.
  - A single event upset (SEU).

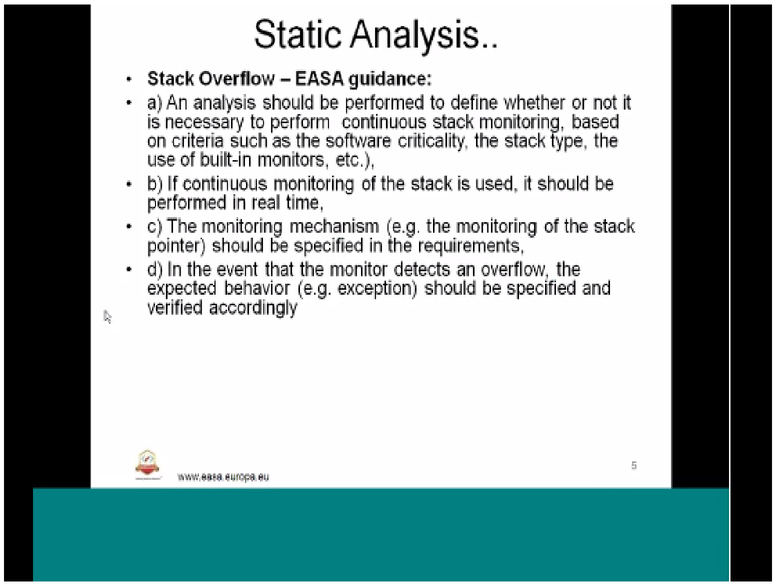
www.easa.europa.eu

3

So an unintended stack overflow may occur during the execution of a program for different reasons. The reasons could be a hardware failure such in the module interphone or the processor etc. a software development there are programmer or developer would have introduced and in own memory of such error to the programmed. That it would be in the software development error.

An unintended software behavior that means software some issue suddenly it is unintended. So, that is one of the reasons. Memory corruption, memory is not written properly. And a single event upset. Due to us stack overflow would have happened. So likewise we have different reasons for this stack overflow. This is to be analyzed.

(Refer Slide Time: 02:32)



Static Analysis..

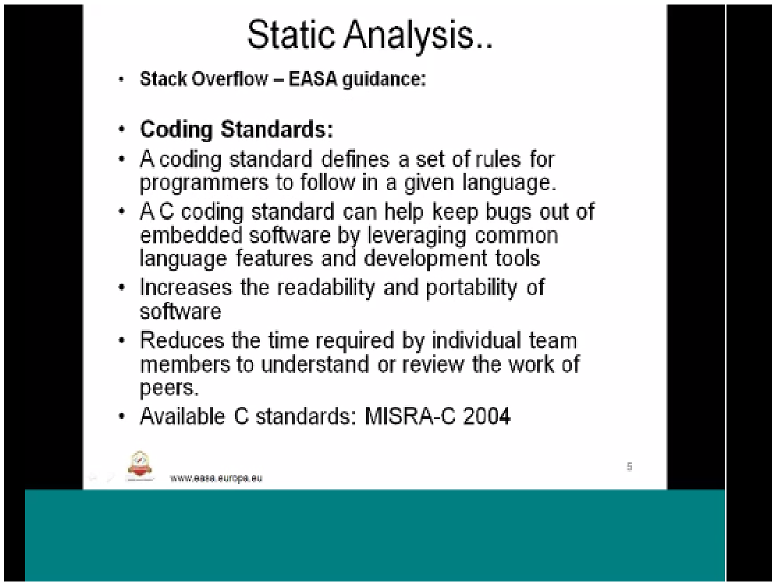
- **Stack Overflow – EASA guidance:**
- a) An analysis should be performed to define whether or not it is necessary to perform continuous stack monitoring, based on criteria such as the software criticality, the stack type, the use of built-in monitors, etc.),
- b) If continuous monitoring of the stack is used, it should be performed in real time,
- c) The monitoring mechanism (e.g. the monitoring of the stack pointer) should be specified in the requirements,
- d) In the event that the monitor detects an overflow, the expected behavior (e.g. exception) should be specified and verified accordingly

www.easa.europa.eu 5

So as per the ESEA guidelines stack overflow need to be taken care such whether the analysis is non performed to different whether or not it is missed it to perform continuous stack that means to building program or not. This continue monitoring of this stack

Is used which will perform in real time when the program is running. It is a specific task allocated for it tiny program or tiny strains of time. That can be used. And we monitoring mechanism should be what the requirements are. The event that the monitoring takes the time so overflow them accepted behavior should be straight and forward. That is what we earlier talk. The next one is the spring's time's needs.

(Refer Slide Time: 03:19)



Static Analysis..

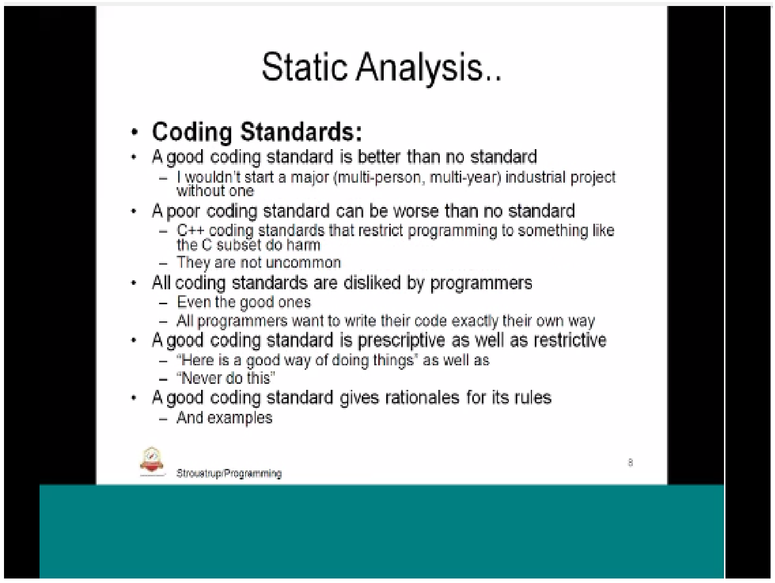
- Stack Overflow – EASA guidance:
- **Coding Standards:**
- A coding standard defines a set of rules for programmers to follow in a given language.
- A C coding standard can help keep bugs out of embedded software by leveraging common language features and development tools
- Increases the readability and portability of software
- Reduces the time required by individual team members to understand or review the work of peers.
- Available C standards: MISRA-C 2004

www.easa.europa.eu

5

Is the one of the important aspect that many will not to or will not follow this will be done by the IVNV the independent validation verification team. In all to check that the codlings have to been followed. So why we codlings standard is important? There are high chances have been done due to interptation are violation of the coding standards. Program can have bugs for all that there are different guidelines and services so one of them have been highlighted here. And it is recommended to you is since standards and tools such as checker and MISRA-C 2004 rules. So, there are different types of standards and guild lines that it recommends.

(Refer Slide Time: 04:15)



Static Analysis..

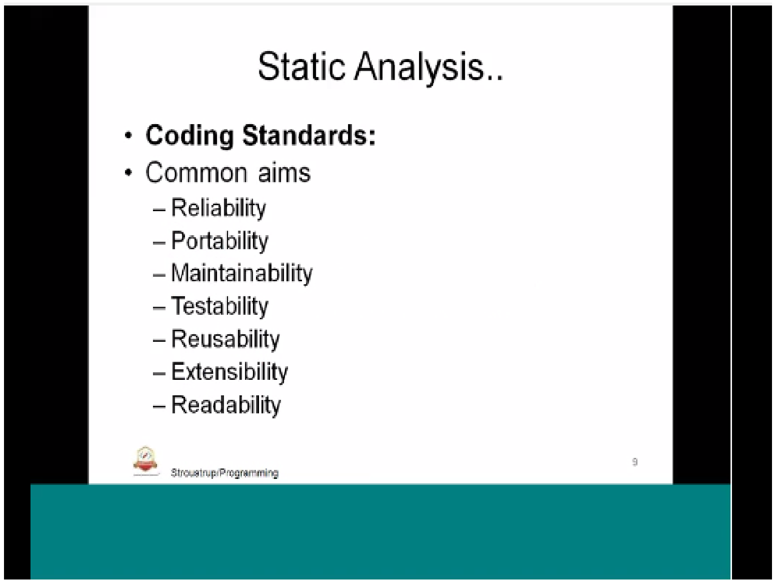
- **Coding Standards:**
- A good coding standard is better than no standard
  - I wouldn't start a major (multi-person, multi-year) industrial project without one
- A poor coding standard can be worse than no standard
  - C++ coding standards that restrict programming to something like the C subset do harm
  - They are not uncommon
- All coding standards are disliked by programmers
  - Even the good ones
  - All programmers want to write their code exactly their own way
- A good coding standard is prescriptive as well as restrictive
  - "Here is a good way of doing things" as well as
  - "Never do this"
- A good coding standard gives rationales for its rules
  - And examples

Stroustrup/Programming

8

Recommends symptoms of the complexity if the program it is poor coding standing can works then more standard so, that it to have a standard.

(Refer Slide Time: 04:26)



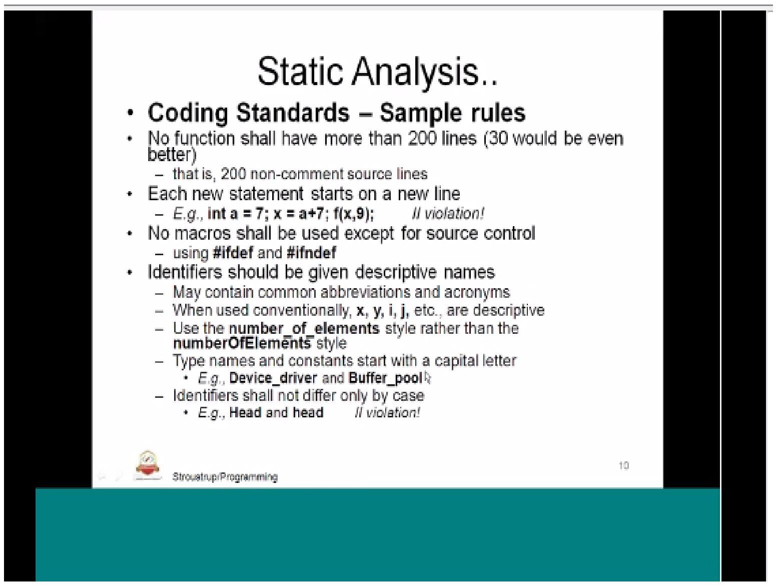
Static Analysis..

- **Coding Standards:**
- Common aims
  - Reliability
  - Portability
  - Maintainability
  - Testability
  - Reusability
  - Extensibility
  - Readability

Stroustrup/Programming 9

There are common object of the coding standards, that reliability, portability, maintainability, testability, reusability, extensibility, readability. That is very important. So, that the code is maintain of them. So sample rules so we had seen.

(Refer Slide Time: 04:41)



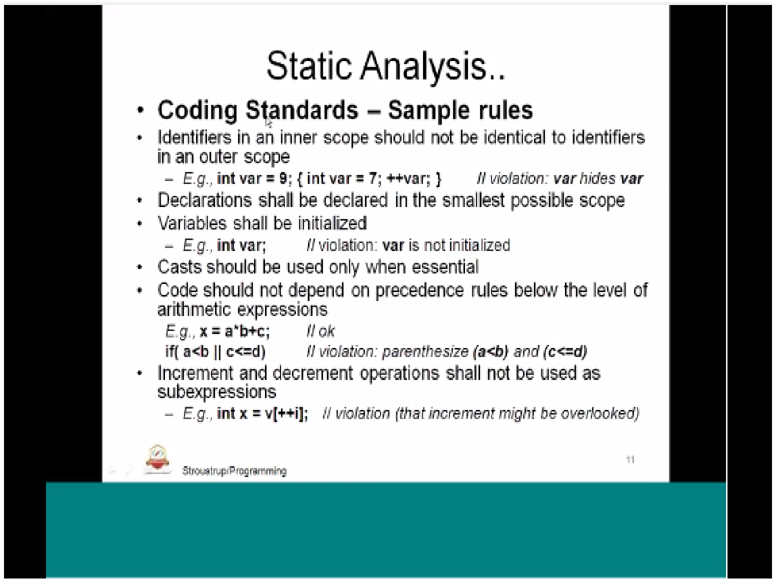
Static Analysis..

- **Coding Standards – Sample rules**
- No function shall have more than 200 lines (30 would be even better)
  - that is, 200 non-comment source lines
- Each new statement starts on a new line
  - *E.g.*, **int a = 7; x = a+7; f(x,9);** *// violation!*
- No macros shall be used except for source control
  - using **#ifdef** and **#ifndef**
- Identifiers should be given descriptive names
  - May contain common abbreviations and acronyms
  - When used conventionally, **x**, **y**, **i**, **j**, etc., are descriptive
  - Use the **number_of_elements** style rather than the **numberOfElements** style
  - Type names and constants start with a capital letter
    - *E.g.*, **Device_driver** and **Buffer_pool**
  - Identifiers shall not differ only by case
    - *E.g.*, **Head** and **head** *// violation!*

Stroustrup/Programming 10

Like we can I have multiples out line for single line red ability is lost and we cannot have macros, except source control, identified should have been forwarded with the descript or reaming full names. And the functional lines cannot we more than 200 lines.

(Refer Slide Time: 05:05)



Static Analysis..

- **Coding Standards – Sample rules**
- Identifiers in an inner scope should not be identical to identifiers in an outer scope
  - *E.g.,* **int var = 9; { int var = 7; ++var; }** *// violation: **var** hides **var***
- Declarations shall be declared in the smallest possible scope
- Variables shall be initialized
  - *E.g.,* **int var;** // violation: **var** is not initialized
- Casts should be used only when essential
- Code should not depend on precedence rules below the level of arithmetic expressions
  - *E.g.,* **x = a*b+c;** *// ok*
  - **if(a<b || c<=d)** *// violation: parenthesize **(a<b)** and **(c<=d)***
- Increment and decrement operations shall not be used as subexpressions
  - *E.g.,* **int x = v[++i];** *// violation (that increment might be overlooked)*

Stroustrup/Programming 11

Did you maintain and it scope of the identifier should be very clear so, that the declaration should be declared in the smallest possible scope. And type case only use the required not always and the precedence is one of the important aspect of the programming. A precedence rules of the controlled. And increment and decrement of the rules could not to be used it is a sub expressions.

These are the one of the rule that we have seen in the earlier class. Today we will see about continuation of the static analysis and code reviews. Okay, so, coding rules power of them it is one of the good guidelines given by Michel mc dowel. He recommends 10 good rules if you follow definitely your program is mature, or Trans of bugs or very much avoided in terms of the trivial issues could have been proven in by the programmer of the coder. Okay, what are those powers of 10? It is tried to understand restrict to simple control flow constructs. That gives control flow construction of as simple as possible try to break a bigger complex programmers into smaller constructs that is what it means. The first rules are about. Does not use go to statements set jump one jump or regression? So it is recommended to provide these things. Second rule, these are called as a golden rules by Michel mc dowel, so second rule is about give all rule to the upper bond.

Definitely, we need to have for loops but with the fixed bond where it is going to complete or where is going to end. For example, we have a loop I less than 0 I plus, plus it is not too be completing here. More on I should end with 10 or 100 or whatever it is. Better to avoid any expressions such as cost and value something like that constant expression. So this will definitely, lead to an upper bond, an upper bond has been using for loops or wide loops. The next golden rules do not use dynamic memory allocation after it is initialized. So, we know initializing is important any variable are objects that are to be defined should be initialized using the initializing function.

Once we have initialized the variable or memory or any program flow constructs. We should not further work on the re initializing of the memory, especially that is what we talked about. So better avoided this kind of a allocations. How much memory is going to take during the programmed instruction? You fix it means initialization of allocate accordingly. So, do not use dynamic memory allocation. But dynamic memory allocation is a big chapter, by itself embedded software system. Some peoples are having memory allocation because they do not know the behavior of the program.

So, especially, where they use speed possible code there is a free new and all that part of dynamic memory allocations available. But it is recommend avoiding that much, and especially after initializing of if it all using the dynamic memory allocations. Do not use again. So that is what it means. Okay, the next one libel functions to no more that 60 lines of text. So you have seen 200 lines in one of the guidelines, what Michal mc Douglas says it 60 lines is the maintenance code.

Or a good enough number of lines for the embedded system function. Text here means an executable some difference of code or executable object code. It is also called a line of code. Lines of code could be having executable plus commands. What it he means here is programs so, better avoided more than 60. Use minimum two assigns for only function of more than 10 lines. So, if function is having more than 10 lines, minimum 2 assigns have to be there that means every function should have 2 assertive statements, minimum, If it is more than 10 lines. That is what was changed.

61 declare data objects and the smallest possible level or called that means we have a inner most available of the scope we are going to defined with the looping function sub expression likewise. Better to declare if it required to be used in that level. So use within that levels of scope any declaration of the data objects. Which is going to hold the data? So you check these rerun value all non complaints. You know void function or non void function. Void functions will write nothing, it is 0. Depends on the system again it could be 0. Non void function will write any non value. So, check the written value of all non void function and check the validity of all function for this. So, it is a golden rule, very good should be applicable or any standards means the parameters, what are you follow or we pass it will function the parameter 1 parameter 2 so like this we have a declaration level parameter N is the declaration. And here we have written

suppose written is some int X could be a better avoid some signals, so suppose written as a signal and the any picture either it is right way, it is getting done the value that it is going to return especially, for where we have a written for some value. It is nothing but non void functions. And also we need to check the validity of all function parameters that means the parameter is going to pass and used in the function better to check it is slightly define and used so the next one is to limit the use of three processor file inclusion and simple macros, that means preprocessor line are there right, and any macros etc.

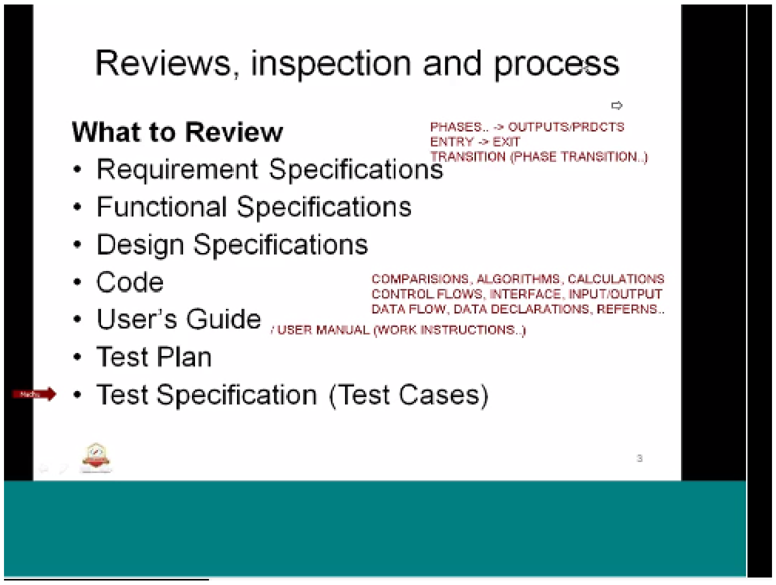
So all this you avoid and use only for file inclusions, includes for example, studio IO.H, and any macro like define, debug could be true. So simple macros is and any inclusions, file inclusions use it or limit it basically, not more than this actually there are number of preprocessors like or, if, all that, try to avoid how simple macros for these case, so it is better inter ability and bugs are less, they having a simpler macros, the next programs may need better documentary trend use with as per the need it need the control of the developer or the re programmer next one in the limit the use of point so point is very important aspects of the embedded system this to a memory, where it is getting a reference.

When it is getting used to try to use as simple as point as in the reference tries to arise variables in all there. When it is absolute is necessary use it. Use no more than one level of reference is for especially. We have the reference here to the point of that it used do not use more than one level. For that, dereferencing of the so limited last one being compile with all one available in pedantic mode and use one or more modern static source code analyzers. That is what source code analysis we have seen we have studied like in terms of understand for C++ are poly space and souse that and try to analysis ay warnings any violence's for that and in additions very important to have the compiler throwing warnings or info or error fixed that is very important. So we need to have the complier giving the warning.

So we cannot disable that option will compile the warning could be a correstatical error, or correstatical issues because complier would not care about that. What about the user as set that it will going to do it. So better to take to all the warnings that is what it is pedantic mode you means, doable it and it one or more static analysis to analyze any dead code is there complexity of intro all that you taken care.

So that is what the 10th rule is. So this is what the classic coding rule. That is to be followed if it followed an appropriately. Definitely, which follows program being having errors is very less? That is what based on Michal McDougall puts is be taking here so that the end of coding rule code reviews.

(Refer Slide Time: 16:32)



Reviews, inspection and process

What to Review

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

PHASES.. -> OUTPUTS/PRDCTS
ENTRY -> EXIT
TRANSITION (PHASE TRANSITION..)

COMPARISIONS, ALGORITHMS, CALCULATIONS
CONTROL FLOWS, INTERFACE, INPUT/OUTPUT
DATA FLOW, DATA DECLARATIONS, REFERNS..

/ USER MANUAL (WORK INSTRUCTIONS..)

3

Now we will come to the embedded software, program reviews, inspection and process. This is very important part of the static analysis this is one of the important topic for embedded software testing. So it have right methods of reviews right inspection sod progress very followed so that daring the review we will bring out all the program or any of the elemental of the embedded software system.

So this is very important to have reviews inspection and process. So each development phases the translation form in the previous class you know. And how you will move between phases we will generate for products right, we have seen the entry and exists here. So each exit criteria we will receive from a work product. So we know that we have embedded systems developed in phases each phase will produces the outputs or any products or dry products from the review. It could be any of the development phases. And this purely based on the translation of the entry to exit.

So as we will talk about exit as an evident we know that some product is been done or delivered. So that is what it means. This transition need to be tested. How we are going to test is this transition is proper or not this is called transition. Or phase transition. We have several that it will come out. That need to be tested in those are something like by documents or coder any of the analysis anything it could be so all these need to be tested. So tested we will do with the help of reviews or inspection of got through there are different types of reviews we will study that one by one.

Okay, so we know that in an early stage of the development cycle majority of all the defects 50% of the all defects comes from the requirements or the functional specifications. And the review process should prevent the defect divagation. The review process we will prevent the defects at the earlier stage. Cost of defect migration to the next page id must higher than finding defect of the phase where it was introduced.

And the next stage, it can cost and the order of magnitude more than and does not to the stage after that. Os we know every stage, we pass on so the cost will go high as we get more issue. So, that the cost is maximized the error is just the structure of the product is deducted after the product is to the customer and minimized with the deducted in the phase where it was introduce. So better to pervilaized effect when it introduced or when it is reflected as you done. And avoid or to reduce as much when it is going to the processor going to the customer. Okay, so we know there are different phases of the life cycle of the embedded software system. Just it could be a requirement specification, function specification, design specification, code, user's guide, test plan, test specification. These are typical embedded software system so we know that requirements specification types the purpose of the requirement phase is to the uses of properly understood.

Before translating the design in the requirement specification the purpose code performance of the requirement is clearly defined expect that thing which performed against the user requirements. So all this hard defects need to be reviewed. The next one of the functional reviews specifications this function design use the process functional specification and designed is the process function specification or design the process of concentrating this requirements initially set of inter face specification or the function specification.

In the functional specifications the scope care test and performance criteria of the system in terms of hardware of and software that may to be the requirements are clearly defined. So system testing is performed against functional specifications. Basically this requirements specification is either testing or accept the testing function specification which we will test of the system level which is nothing but system processor.

So definitely there are products are buy products of these specifications that means too high a review or analysis. The next is the designs specification. So the internal the design of the process is system requirement into the detail set of instruct their data structure and data flows algorithms anything it could be. So in design specification the more power of the typical solution of posing against the exits architecture and application to need the system requirements of clearly specified.

Compound testing is performed against the internal design. The architecture design or low level design is also called as and the component testing is with the help of the code. Also it will use the base with the help of low level design. The next is the code. That is the process the code is the process of translate the internal design or specification low level internally specific set of code. So what we are going to check.

Code is references. References there are data declaration issues, computation issues, comparison issues, control flow errors, algorithms, calculation, control flows then what else interface errors, input, output error, data flow, data declaration, references, all these are the part of the code and check against that should be done when we do the review of the intended code the next one so the user's guide. So basically every software products that is delivered to a customer is the both executable code and the user moment so basically executable binary they going to deliver. And how to install and how to use the embedded software code on the target board has to be instructed properly to the user of the system.

This is nothing but user manner. It is also called as you should be having the work instructions how to use this. So this document is known as important manual code. So this code is important like code it was it is quality is there significant factor the success of the success or failure of the product. From the point of view of the user if the manual says something and the user follows this functions and if does not works the really helps that the code is instruct is right, that means some issue it is not been behaving what is the documented. So that is what been behaving what is documented.

What is the document of the plan is required. Test where we know lot of the aspects in the test plan. It has to take all these and who is going to test what is schedule what is the environment all these will be part of the plan. Like tools coasted I mean the schedule all are going to take what sort of a level of testing which going to become it help what tools likewise it is going to have. Last one being the test specification of the test case instructs. Test cases are strategies all is going to be designed, all is will be there that is to be reviewed or inspected using a process. That is what the review has to do.

So these are the things that have to be taken care doing the review process across the life cycle of the project. So basically why it is called, each life cycle will produce some sort of the output. The output could be one of these requirements function specification, design, code, users guide, test plan, test specifications.

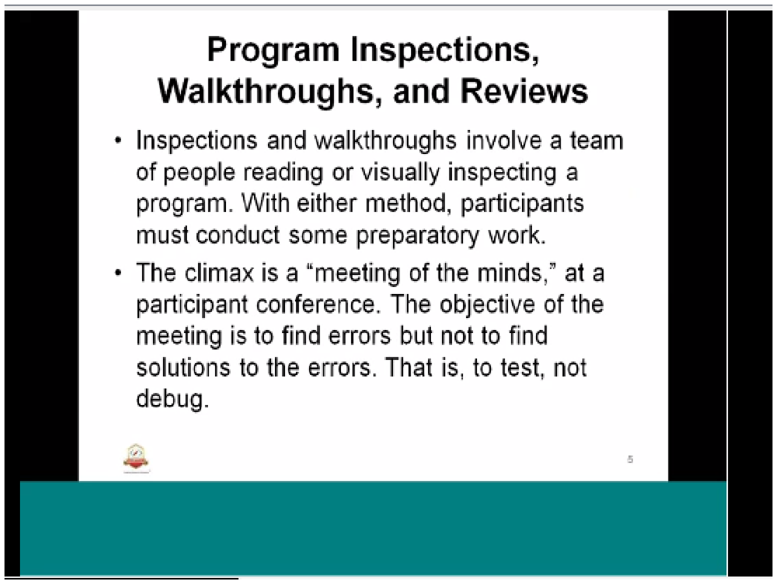
Of course you can have more test reports, execution reports, process, metrics, etc. so, you will take each one when you go through the next session process. Okay, the next one is how much we can review. So the first one is what to review? The embedded system, the next one is how much you need to review? I am going to review the entire requirement and I am going to review all the metrics that are support the information for the requirements likewise. We are going to have so we need to have a careful sampling it is called as sampling basically. Suppose for example, these are specification these are the functional specification, we will going to sample using the sampling criteria.

You should be defined during the plan of the testing of the development. Which it says some supports 30% sampling which should be done that means there are thousand requirements 300 requirements, how to be sampling. You see that whether it is implies appropriately, with in the assigned, or in differ module, as per the specification is as per the standard likewise. This is basically from independent prospects.

Of course the development perspective that ever is going to develop has to do a 100% review is called as self review. Self review has to be 100% and we have a clear review that is also should be a 100%that 50% review means person A product will be reviewed by person B and person B is sort of implicated by person A. so both of them are peers. To have an independency they need to cross that against each outputs of the products so all these like designs specifications, code, and code has to be reviewed, user's guide has to be reviewed, in terms of sampling at the different levels.

Of course we have a quality in a quality person is going to review based on his criteria based on it could be in 10% 20% is again depending on the criticality of the program. Program is nothing but the entire life development cycle or criticality of the level of the software. Whether it is more critical safety contents are there security is all this chapters in terms of the review selection and that is what the important for doing the how to review aspects.

(Refer Slide Time: 30:14)



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

Okay, coming to the review detail like, we have different types of reviews and different types off program inspections and different types of walkthroughs. So what are those? Oaky and why it is important? Why because not all testers of software applications reads decor the concept of the studying program code as per the testing effort. Certainly is accepted. Because t is going to be emphasize as a one phase of program inspections walkthrough and reviews. Definitely we had to have a dedicated process for doing these.

Then only the important of the understanding the code and reviewing the code will going to exist so several factors may affect the likely given testing and given actually reading the program code the size or complexity of the application size of the development team. It time line of the application development. All this matters, and the background and the culture of the program and the test in team all these are very important in identifying the review process walkthrough the instructions all this are important factors.

So basically this is not a computerized based testing. Of course we use some tools in terms of identifying the reviews walkthrough set of that. But this is not automated. Generally the reviews are not automated. Surely it is done by humans for human testing or you want to call. So no automation is done. Here automation is not done means for reviews there is no tool in terms of producing the review aspects. But once we have reviews we can have the metrics and category of errors and all that using the excel sheet.

Or any tools such as maxilla whatever it is. So, human testing is basically called and inspections and walkthrough in team of people reading or visually inspecting the program. With either method, participants must conduct some preparatory work. Any method they can use it. But you should read. Or you should visually inspect. That is what inspection means and walkthrough also same thing. The climax is a meeting of the minds at the participant conference. Then the objective of the meeting is to find errors but not find solutions of the errors. That is to test not debug.

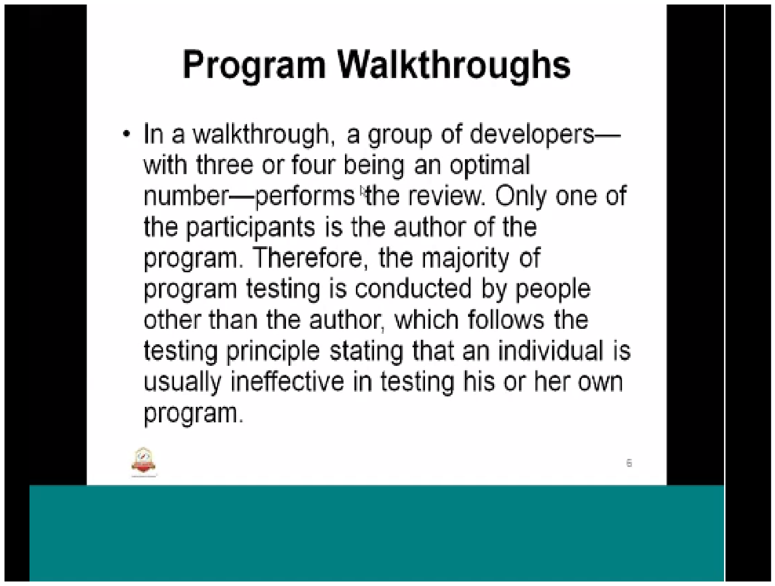
The primary thing that walkthrough the inspection does use first they will go through the or there inspect the program. It could be a requirement or it could be a code or an execution anything the part of the embedded system. So it is very important to have process in terms of inspections of walkthroughs and basically at the end of their going to greet all the reviews we were done all the inspectors we were as the entire participations comes us the objective is to be find the problem. That is all and concludes whether the problem is rightly identified or wrong or not rightly done. But there is no solution that is going to be identified. Just reporting have been inspected or that is been reviewed.

That is what the aim of the next phase of the inspection of walkthroughs. So there are different methods for doing this that could be any analogy method such as standards following or any program process or we can have some criteria reading in the computer based testing. That means we can have a checklist identified in a tool or excel sheet and against checklist that will be a tick mark user will keen and to generate the report.

So that is a about program it could be a requirement a code it could be execution, anything. This is part of the aspects of the embedded system. So that is very important to have process in terms of inspections of walkthroughs. And basically end of that they are going to weight all the review as we were as done at participates of this the object is to find the problem that is all. Conclude whether the problem is rightly identified or not rightly done. But there is no solution identified just the reporting of the errors.

That is been inspected or that is been reviewed. That is what the aim of the next phase of the inspection of walkthrough. So there are different methods for doing this that could be any analogy methods such as any standard following or any programming process or we can have some criteria in computer based testing. That means we can have a checklist and defined in a tool or and again checklist that will be a tick mark and non tick mark user will keen and will generated in the report. So that is about program inspection walkthroughs and reviews this study about program walkthrough.

(Refer Slide Time: 35:42)



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

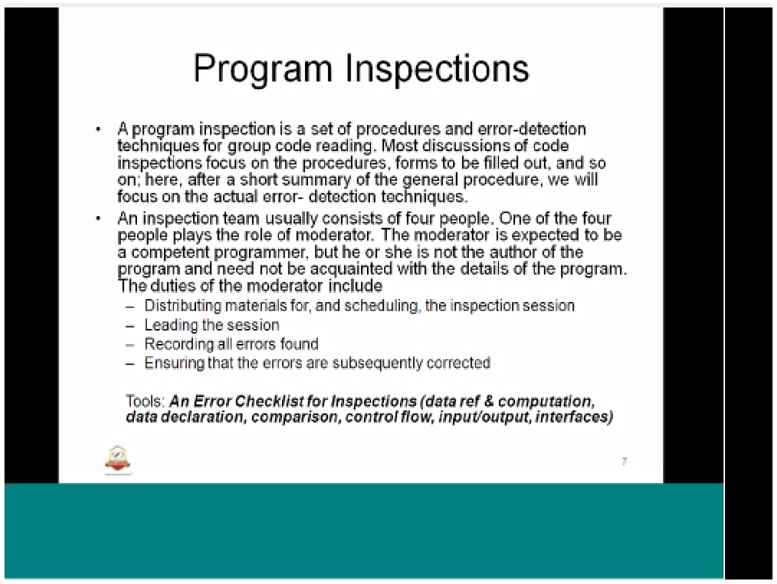
Okay, so basically it is refer form of part of software testing. What is says is a walkthrough, a group of developments with 3 or 4 being an optimal number it could be more depend on the complexity of the embedded system. They will perform the review. Only one of the participants is the author of the program and majority other than the author. So one will have a author and the other one will be supporting in terms of during of review.

So they will do following the testing principal stating that these usually ineffective in testing his or her own program. So basically we do in a walkthrough a group of developers 3 or 4 depending on the progress one will be the author other will be he reviewers. So that is what going to happen.

So an inspection of walkthrough is an improvement over the holder test checking. Usually developers used check by self review this is an improvement that we have seen now a days. Inspections and walkthroughs are more review basically again because people other than the programs authors are involve. Definitely it is going to be effective. The independent and UN biased reviews aspect to the taken up other advantage of walkthrough resulting in lower debugging. That means either corrections anything we need to do.

We do not have much time the cost will be less. And when there is a error form it is can be located which sort of the code this error id there. This is the advantage of this inspection. In addition this process frequently exposes the batch of error or a group of issue fallowing error to be corrected in one sort. Whereas inspection if you do a computer based testing, normally it exposes only symptoms of the error. It will just error it will not well defined the problem is. So that is where the advantage of inspection is there.

(Refer Slide time: 38:26)



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

Okay, so inspection continuing that the program inspection is set of procedures and error deduction techniques, for group code reading. Basically it will be read in a group. More discussion of code inspections focus on the procedures forms to be filled out, and so on. That means there is a checklist and review guidelines as per the inspections process will be taking care. Here after a short memory of the general testing here we will focus on the actual error deduction technique.

An inspection team actually consist of 4 people one of the 4 people is basically it is an example of typical embedded software. Do not get bias take business and defined number it could be 5, 6 different on the complexity and the intended program. The less also so one of the 4 people plays the role of moderated. The moderator these expected to be a combatant programmer but he or she is not the author of the program. Basically he coordinates moderate and he need to be appointed with the details of the program you knows whom to contact who is take over there? All these information the moderator will have.

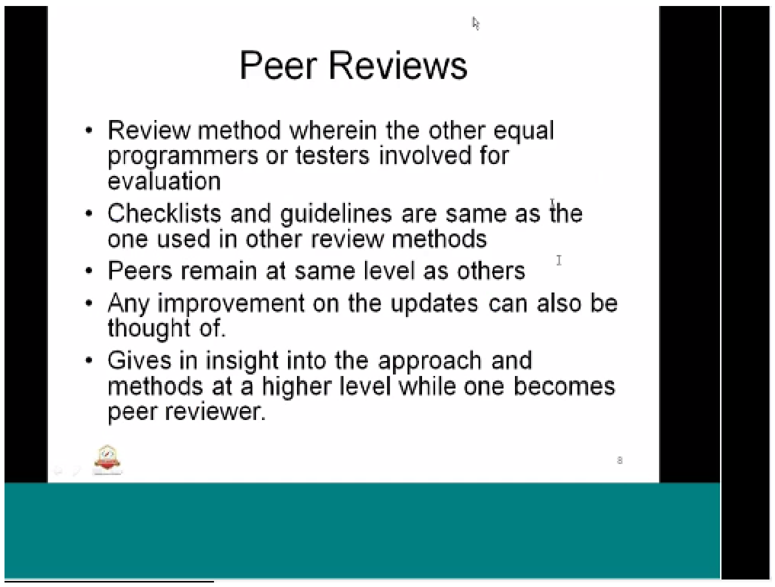
So the duties of the moderator will include distributing materials for the people and scheduling the inspection and the inspection session. And he should lead the session. The record all the errors static form in the session I assuring that the errors of subsequent and he should lead the session. You should record all the errors that is formed so that the errors that are going to be clear are not will be communicated to the errors. Okay, tools an error checklist for inspections data represent compactions comparisons all it will be part of the checklist. Against that the check of the programs will be done. So basically the moderator is like a quality controlling in a second team member is the programmer the remaining team members will be the program designer or the different programmer.

And test specialist also can be involved. The moderator distributes the program list and designs specification to be other participants, well and advanced before meeting before the inspection session. The participant is accepted to familiarize them to the material required the material. During the session basically couple of activities will the programmer narrates the statement logic of the program, so Suring this course other participants should raise questions. And they should be perused to determine whether like this or not.

It is like the programmer than the other team members we will find many of the errors on during the narration. They will raise the hand and raise the issues the simple act of reading alone a program to the audience seen to be a remarkable an effective a detected technique. This is what one of the activity. The second activity the program is analyzed respected to the checklist of historically common programming errors. So two things will happen one is narration, other one is analysis. So narration he will narrate code logic and that entire so user will raise the doubts then it will be recorded.

The second one analysis is will be done against the checklist such as what are to be listed here the other program has any errors symptoms of data represent computation set the so always will be recorded in the session. And the result of the inspection he will distribute the errors the developer or the author of the program. And the moderate has insured that those errors are that I fixed or justified or it is rejected or accepted by the implementer. That is what he is to be ensured. Okay, that is what program inspection is.

(Refer Slide Time: 43:28)



Peer Reviews

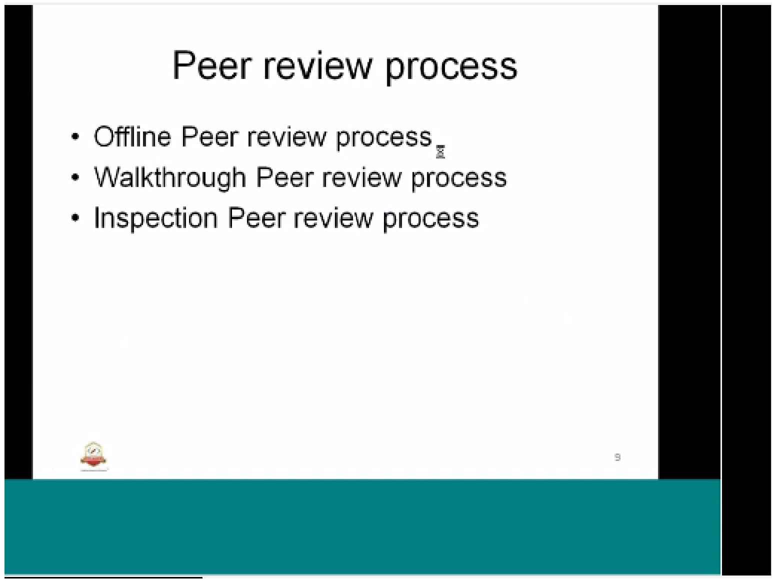
- Review method wherein the other equal programmers or testers involved for evaluation
- Checklists and guidelines are same as the one used in other review methods
- Peers remain at same level as others
- Any improvement on the updates can also be thought of.
- Gives in insight into the approach and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

So next one is the peer reviews. The peer review is also one of the important embedded system static analysis review method, so what are they going to do in a peer review. And we also study about peer review and types of peer reviews so peer review method warning the other equal programmers tester involve for peer review. Other set testers are programmers are independently reviewing each other at the test. That is what peer review above. So there is again some guidelines same as that are used in the other review methods. Peers remain the same levels of that. So that is why it is called as peers that means there are some 10 functions and 10 functions are grouping the developed by 10 people.

And for these 10 functions is above and another 2 people are developing the test cases or test functions all are considers as peers and they can review across their auditors so they are called as peers. Any improvement on the update can also sort of. That means it is not just enough to do the peer review identifying the bugs of identifying the errors that program or test has the also important to have an improvement in terms of optimizing automating and speed I will say improvement in terms of effectively effectiveness. In terms of the output of the program how much it can be effectively done.

It could be a pervious method, it could be short method all is can be sort of. So that is what improvement can be expected in the peer review. So gives an insight in to be approach and method with a higher level, while one become peer reviews that means it gives a polis tic approach of what is intended the peer review fact that is what it means. So peer reviews is very important aspects we will see peer the process.

(Refer Slide Time: 46:21)



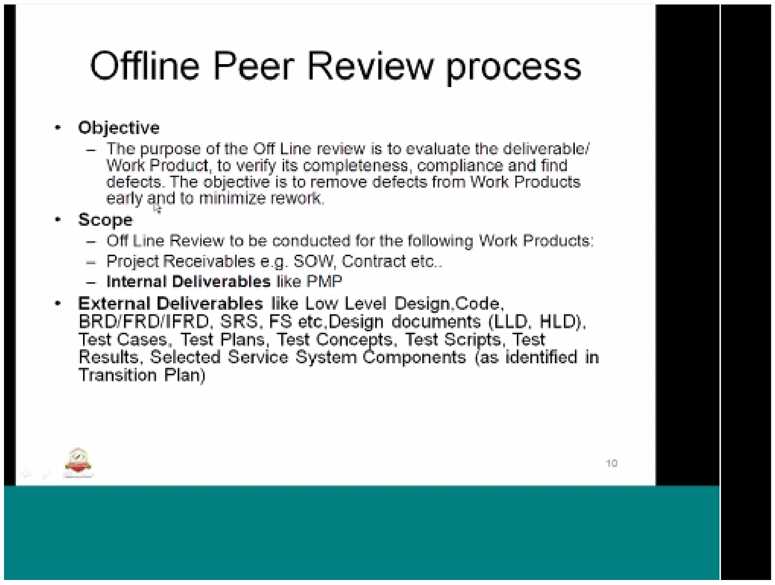
Peer review process

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

9

What is the peer review process? And types of peer review process. So there are 3 types of peer reviews can be done, offline peer review, walkthrough peer review, inspection peer review what are those?

(Refer Slide Time: 46:34)



Offline Peer Review process

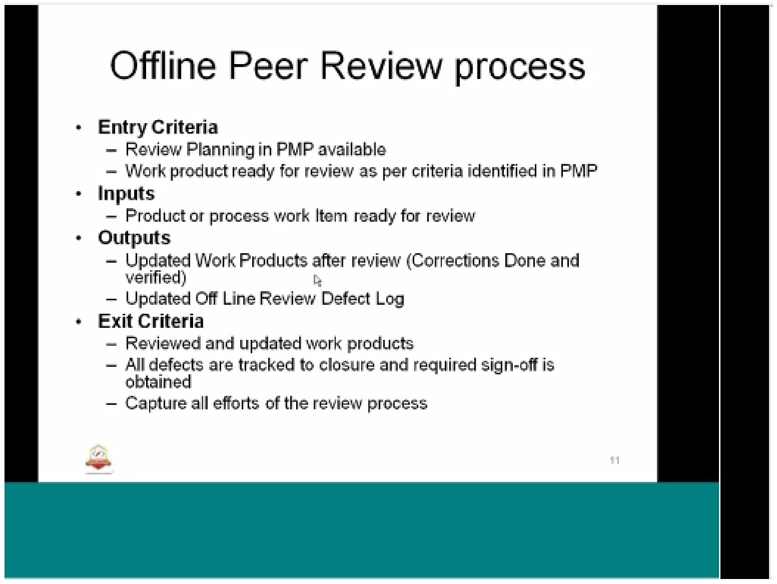
- **Objective**
  - The purpose of the Off Line review is to evaluate the deliverable/ Work Product, to verify its completeness, compliance and find defects. The objective is to remove defects from Work Products early and to minimize rework.
- **Scope**
  - Off Line Review to be conducted for the following 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

Okay, offline peer review process is as per the objective the purpose of the off line review is to evaluate the deliverable work product, to verify its completeness and find defects. The object is to remove defects from work products early and to minimize the rework. That is what the off line peer review. Where peer reviews will not do together but they will take off line offend and they come you with the reviews. That is what the objective. Scope is off line review will be conducted for these following work products.

Basically project receivables statement of works contracts whatever it could be and that is what the off line reviews to be follow. This one example that what we have contracts whatever it could be and that is what the offline peer review is. This is one of the examples of what the output and internal deliverables such as PMP project management plan basically so these all also can be off line review or either result of off line review, process we have a external deliverables. Like low level design, code it could be different such as SRS function specification document such as LLD, HLD test cases test plans, test concepts, test scripts, selected service result component, which are identified in the transition plan. All is part of the off line peer review process.

(Refer Slide Time: 48:24)


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

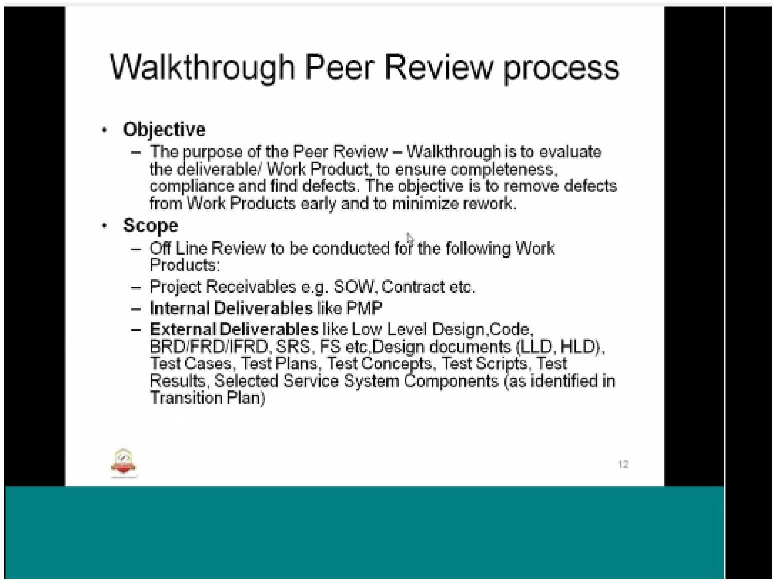
11

Okay, let us see the next offline peer review process such as entry criteria how we are going to enter into offline peer review example is project management. Hoe it is going to be offline peer review. So review planning with PMP level work product ready for as per the criteria identified in PMP. Inputs product or process work item ready for the outputs updated work products review that means review is considered as complete only when the review are taken then and verified updated offline peer review defect log identified by reviews and each defect log there is status such as open flows in progress.

Defect log will have status all could be rejected also. You should show one of the status the defect log finally you make sure that the review record is closed with all the individual defects are closed. So, when it is open that means that defect has been identified and if it is in progress means the efforts of the particular review is view in progress that will under fix. It will not fix. Rejected means that defect identified is not accepted. It is rejected. So the exit criteria of the offline peer review are reviewed and updated work products. All defects are tracked to closure and required sign-off is obtained.

Capture all efforts of the review processes that mean how many efforts w taken in terms of review processor offline is getting capture. Next one is the walkthrough peer review process.

(Refer Slide Time: 50:39)



Walkthrough Peer Review process

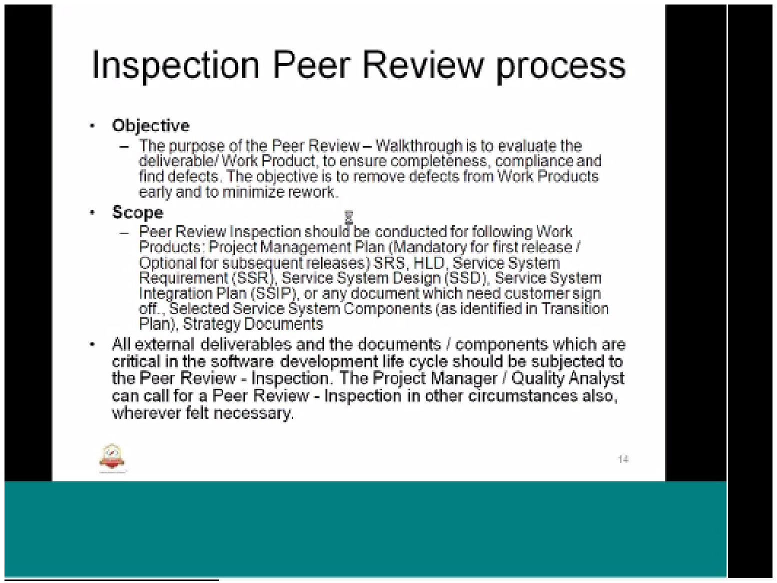
- **Objective**
  - The purpose of the Peer Review – Walkthrough is to evaluate the deliverable/ Work Product, to ensure completeness, compliance and find defects. The objective is to remove defects from Work Products early and to minimize rework.
- **Scope**
  - Off Line Review to be conducted for the 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

we are having to a walkthrough of the hard effects peer review process objectives e is the purpose of the peer review walkthrough it is evaluated the deliverable product to ensure completeness compliance and find defects. The objective is to remove defects from work products early and to minimize the rework. Scope is offline review to be conducted to be following work products.

Basically statement of work, contract documents and it is an example and internal deliverables such as project management plan, the external deliverable such as requirements hardware requirements, functional requirements world level requirements designed documents LLD, HLD test cases test plans and walkthrough peer review process entry, inputs, outputs, exist criteria, same as same whatever we have seen in the offline review process. Here the hard defects are walkthrough with different team members for the peer review as and similar to the instruction and inspection peer review process.

(Refer Slide Time: 52:03)



Inspection Peer Review process

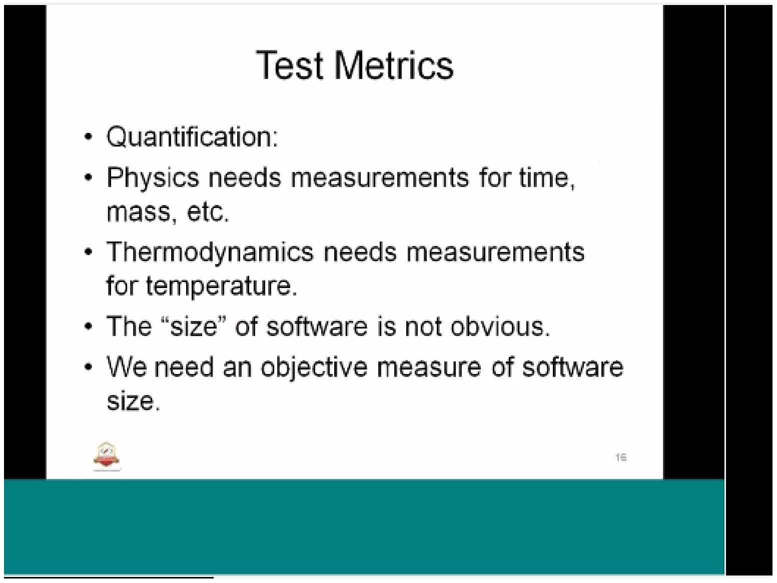
- **Objective**
  - The purpose of the Peer Review – Walkthrough is to evaluate the deliverable/ Work Product, to ensure completeness, compliance and find defects. The objective is to remove defects from Work Products early and to minimize rework.
- **Scope**
  - Peer Review Inspection should be conducted 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

What we do the purpose here is to walkthrough and evaluated to the deliverable work product ensure completeness compliance and find defects. The objects are to remove defects from work products early and to minimize rework. Scope is peer review inspection that should be conduct from the work products such as project management plan SRS high level design, system requirements service system design any of the work products of embedded system for the example taken from one of the embedded system process that is followed by the industry. All external deliverable and the documents and components which are critical in the software development life cycle should be subjected to the peer review inspection.

That means all the exponent deliverables should be inspection peer reviewed. That is what the meaning of. The project manager or quality analyst form call for a peer review inspection in other circumstances also wherever it is nesscesry. Here also we also entry criteria inputs outputs exist criteria that have to mention in the inspection theory process. Okay, that is the end of the review process and that is static analysis aspects in terms of the peer review and the static analysis sorry coding standards what we have followed to be last and today’s slide.

(Refer Slide Time: 54:05)



Test Metrics

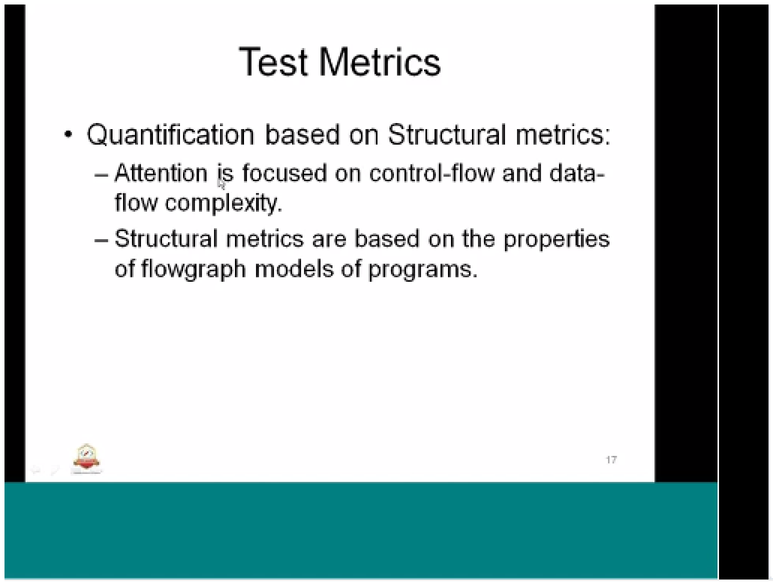
- Quantification:
- Physics needs measurements for time, mass, etc.
- Thermodynamics needs measurements for temperature.
- The "size" of software is not obvious.
- We need an objective measure of software size.

16

The next type of static analysis is test metrics. How all we go to produce the test metrics? And what are the test metrics that are used or embedded software testing and how it is going to be use in terms of static analysis. That is what the test metrics will bring it out. Okay, quantification, so basically test metrics bring or it will quantify the various hard defects in terms of how much and what is the percentage what are the pass fails what are not applicable what the burn down charts is likewise.

We have several metrics. Basically the test metrics are getting quantified. Physics needs measurements of for time mass etc. thermodynamics needs measurements for temperature. The size of the software is not obvious. We need an objective measure of software size. It is not the metrics that is based on the size. But it is objective measurements basically that is what is the quantification. This is very important when we do the metrics accumulation. That means when we collect the metrics we need to be definite about how we are going to collect the metrics. Metrics is not just a sign it can get directly help out the tool how much size, how much lines of coordination are there likewise, so better it will be a determine measurement of the software, that is very important.

(Refer Slide Time: 56:09)


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.

17

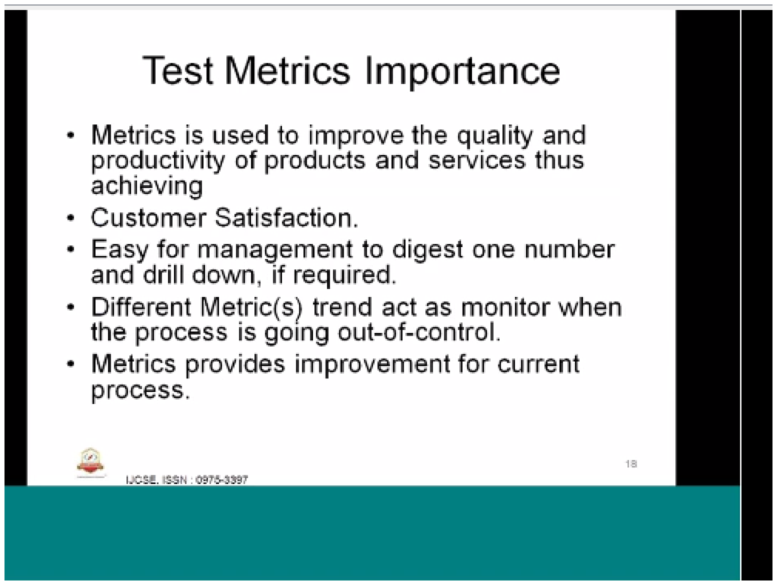
Okay, quantification based on the structural metrics, it not the lines of code as I said it is the structural metrics. So that is what test metrics we will bring it out. Attention ios focused on control flow and data flow complex. We are seen that control flow and data flow the test case execution of the control flow path and the objectives all these will be part of the metrics it will cover.

Structural metrics are base on the property soft flow graphs models of programs. So that is what basically the structural metrics is talk about. So that software quantification we do not on the lines of code. It is basically we need to a motion of the so it is when comes the two testing strategy. When we doing a testing how we are go9ing to test? Basically not base on the complexity and its structure of the code. So something like we may come of this strategy such as strategy A X needs some 2 test for unit. Unit could be a piece of software. So likewise we need to have quantification.

So basically we need to be having questions, before we develop the metrics in terms of quantification how it we test it and what is the strategy and when we can stop the test and how much we can find the bugs. These are something like percentage are it could be some numbers say suppose we have thousand lines of code and we as a tester based on the design requirements and all. We will work out the strategy and he is expected to bring so 10% of the lines of codes. Suppose are may be 2% 10% is too much. So 2% is something like 20 lines. The code is another issue so something like how many bugs we can expect from the piece of software when it is tested.

So there is one of the test metrics. So which testing techniques are more effective? Testing hardly or smartly do we have a strong program or a week test shoot? So this kind of a question has for collecting the various metrics while doing the test. So it difficult answers some of these questions satisfactorily that we need to have a metrics in progress for that. We can discuss the aim of empirical law of symptoms of size number of bug's number of test strategies of expected number of test required to find a bug. Testing techniques it effectiveness all is will be done with the test metrics. This is the basically weight age so that is what we do when we do the test metrics. This is basically weight age, structural metrics, contification you saw we are to take care when we do the test metrics. So it is not we are going by the size of the software. When we capture the test metrics it is going by the test strategy the structural aspects of the software. The structural aspects could be control flow data flow and complexity of the program likewise.

(Refer Slide Time: 1:00:32)



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

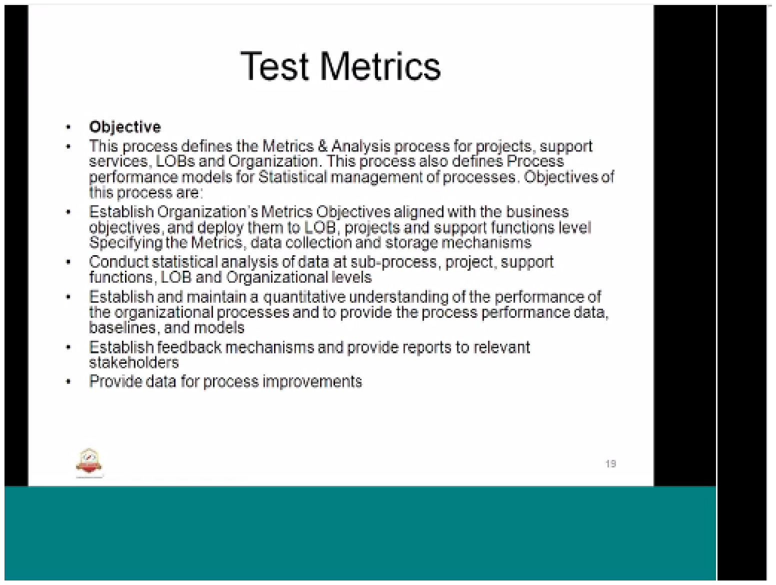
18

So importance of test metrics, why we need test metrics? Metrics is used to improve the quality and productivity of products and services thus achieving. Basically test metrics with the help of we know how much improved for a period of the embedded software life cycle. Or a regression test or different types of test how much bugs we are able to gather how much bugs we are going to fix what is the maturity of the products all the defects come out without the metrics we cannot satisfy be maturity of the product. Definitely we are going to r3eport to the customer. And we can have the customer satisfaction based on this metrics.

And it is easier to manage or easy to management for digest one number and drilled on if it is required. There is a critical issue and they critical issues are getting this is all based on the test metrics. Different metrics and act as a monitor when the process is going out of compound. That means when we do the testing we are come up with they are going to come up with the different metrics and trend of that metrics we will definitely give that is the program is going out of control or within the control.

We will study about this different metrics trend and all that. There are different charts, in future sessions. Metrics basically provide improvement for current process that means when we follow the process it is going to be continuously improved. That is what the metrics is going to help. So basically any process we take such as they will 5 or any level 6 etc. so they basically look for the improvement and minimization of the errors when doing a production when we do a process followed when we are going to have the life cycle implemented where the importance of test metrics is going to come up. Okay, so we understand that the important if test metrics. How we are going too represented. So we study those aspects of the next class.

(Refer Slide Time: 1:03:03)



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

Object of the test metrics and test metrics how it going to be depicted what is robontom aspects of the test metrics how the trends are etc. in the next session.