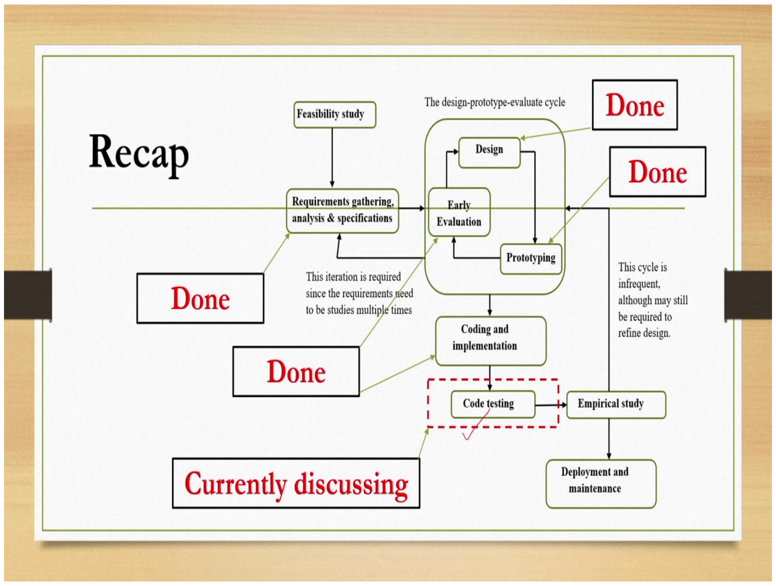
Design and Implementation of Human – Computer Interfaces

Prof. Dr. Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture: 29
Review-Based Code Testing

Hello and welcome to NPTEL MOOCS course design and implementation of human computer interfaces we are going to start lecture number 26 where we are going to continue our discussion on testing particularly we will focus on review based code testing before.

(Refer Slide Time: 01:02)

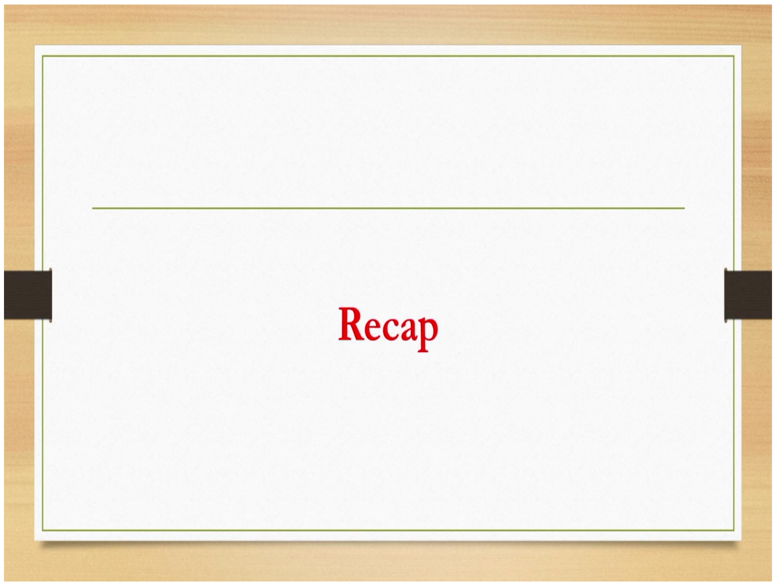

Recap
Feasibility study
Requirements gathering, analysis & specifications
The design-prototype-evaluate cycle
Design
Early Evaluation
Prototyping
This iteration is required since the requirements need to be studies multiple times
Coding and implementation
Code testing
Empirical study
This cycle is infrequent, although may still be required to refine design.
Deployment and maintenance
Done
Done
Done
Done
Currently discussing

We begin our discussion let us quickly recap what we have learned where we are. Now and what we are going to discuss to put discussion in the proper context. So, we are discussing interactive system development life cycle in the interactive system development life cycle there are several stages among those stages we have already covered many of the stages and we are continuing with the remaining stages.

So, we have already covered requirement Gathering analysis and specification stage then we covered design prototype evaluate cycle and the substages that are part of this cycle including design, prototyping and evaluation. Now these three substages are primarily meant for interface design also we have discussed code design. Then we discussed coding and implementation stage where we learned about good coding practices and what we should do and what we should avoid while converting our system design into into an executable code.

Currently we are discussing the testing stage which is located here. In the testing stage we have already started our discussion on the basic concepts and we are going to continue that discussion to cover the broad categories of code testing.

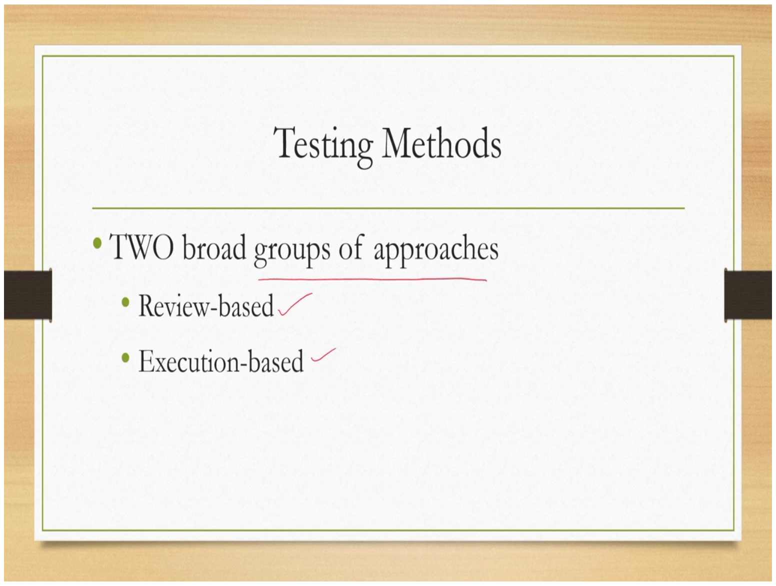
(Refer Slide Time: 02:45)



Recap

So, in the testing stage what we have learned in the previous lectures let us have a quick recap.

(Refer Slide Time: 02:52)



Testing Methods

- TWO broad groups of approaches
  - Review-based ✓
  - Execution-based ✓

We have learned that there are broadly two groups of approaches which we can employ to test our code. Now there are two things when we are talking about interactive system development one is how it is going to be perceived by the user and accepted by the user that is primarily determined by the usability of the design or rather the usability of the system. For ensuring usability we have already seen that during this design prototype evaluate cycle we make prototypes.

And get those evaluated by expert users to know about the usability issues that may be there with the interface design. However so, far we have not talked about the other aspect of a system that is how good the system behaves during execution or rather the efficiency of the code. How well the code is written and how well the system design is implemented that part we have not discussed. So, far that comes under the testing phase or code testing phase.

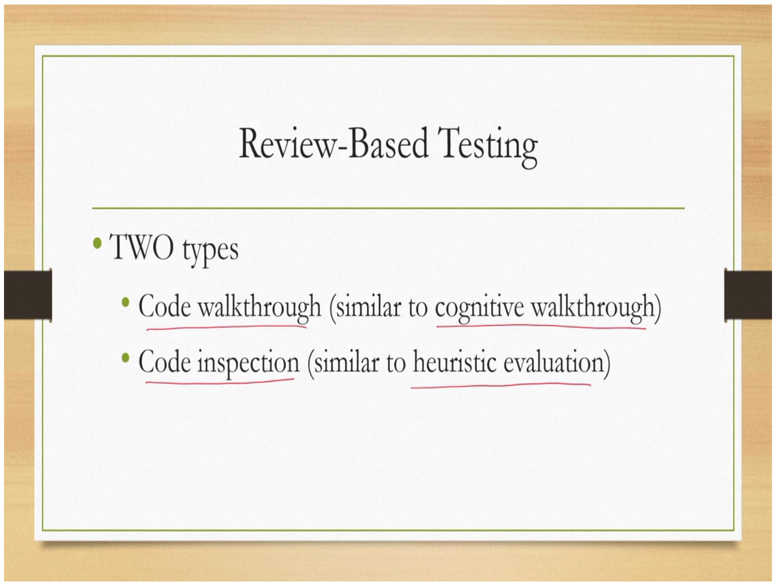
Now there our primary concern is to see that the code is error free. Now errors can happen at multiple levels we may think of errors at the syntactic level that means whether we are following the correct syntax while writing the code using a particular language. We may face errors at the logical or semantics level that is whether there are any logical bugs that are there that are present in our code. Also it may happen that we may fail to follow standard coding practices that is also a kind of important errors or violation of conventions that we should be very careful about.

Otherwise the code that we write will be very difficult to understand by others who may be part of your team. So, that will in turn affect the team work and that is not a good thing when you are part of a complex interactive system development team. To determine syntactic errors of course we rely on the compilers maybe all of you are aware of that we have already mentioned this point in the previous lecture to determine logical errors as well as to determine errors that may happen due to the improper use of convention we take request to several methods.

Now all these methods can broadly be divided into two groups of methods or two groups of approaches one is review based testing method and the other one is execution based testing methods. Now review based testing methods are as we have noted in our earlier lectures quick cost effective way to test your code. Here primarily we try to identify major errors without spending too much time or effort on analyzing the code.

In execution based testing method this is not the case it is more formal and more rigorous in nature where we mostly focus on analyzing the code internally as well as externally and accordingly we design test cases and then go for testing that is a more systematic scientific rigorous and detailed way of testing.

(Refer Slide Time: 07:05)



Review-Based Testing

- TWO types
  - Code walkthrough (similar to cognitive walkthrough)
  - Code inspection (similar to heuristic evaluation)

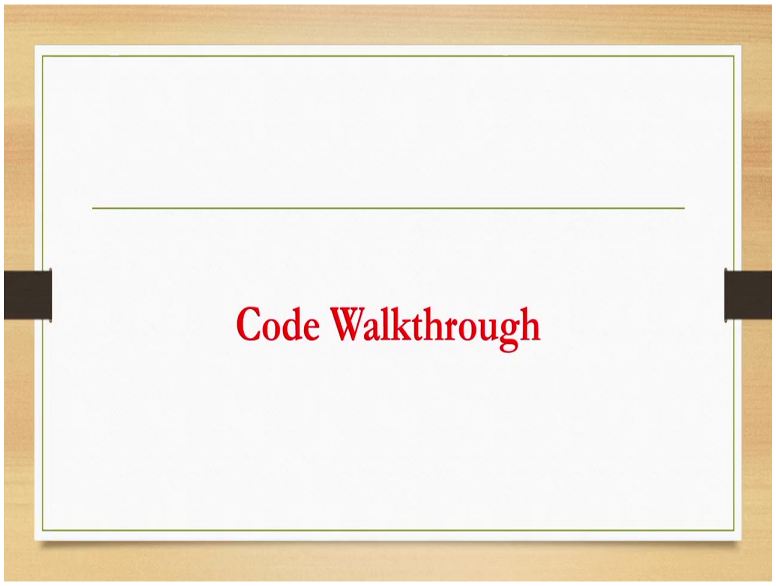
The focus of this lecture is review-based testing. As we said as I mentioned review based testing is primarily used to quickly identify major issues or major errors in the code. So, that those can be rectified and we have to spend less effort and time for more formal testing of the code. Now review waste testing can be of two types one is code walkthrough. Now this is similar to the cognitive walkthrough method that we have learned earlier while discussing evaluation of prototypes.

What we have discussed there that in cognitive walkthrough we have a small team of evaluators between three to five members each member is given a use case scenario and they are asked to identify usability issues while trying to use the Prototype for the particular use case given at the end each of them produces a report which the evaluator combines and determines the major usability issues.

Code walkthrough is similar in nature as we shall see soon. The other type of review based testing method is code inspection. Now this is again similar to another prototype evaluation method that we mentioned namely heuristic evaluation. So, in heuristic evaluation what we did we have a checklist. So, we used Nielsen's 10 heuristics as the checklist and again there will be a small team of evaluators three to five members.

Each of these members will be given a prototype and the checklist and they will be asked to identify which of the items in the checklist are satisfied in the interface represented as a prototype. Same idea is applied for code testing also which we call code inspection. So, conceptually Code walkthrough is similar to cognitive walkthrough method of evaluating prototypes and code inspection is similar to heuristic evaluation method of prototype evaluation.

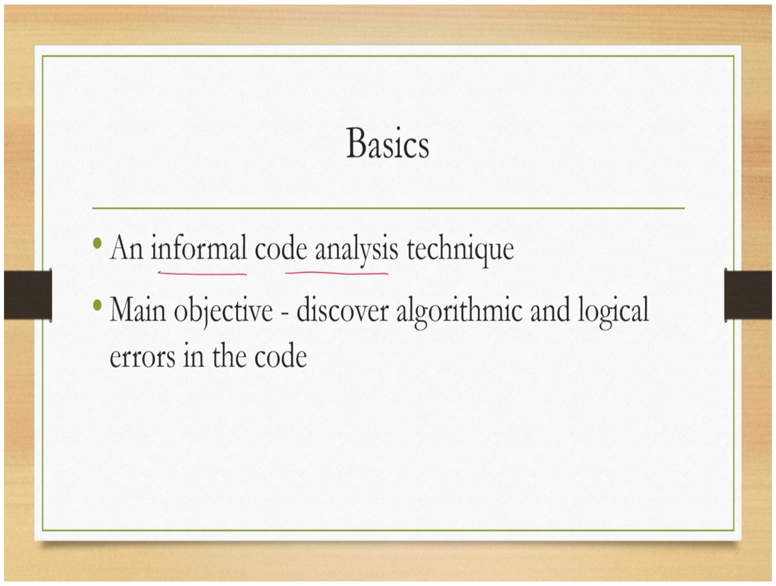
(Refer Slide Time: 09:51)



Code Walkthrough

Let us briefly try to understand the code walkthrough method what we do and what we get at the end. So, Code walkthrough is an informal method for analyzing the code.

(Refer Slide Time: 10:05)

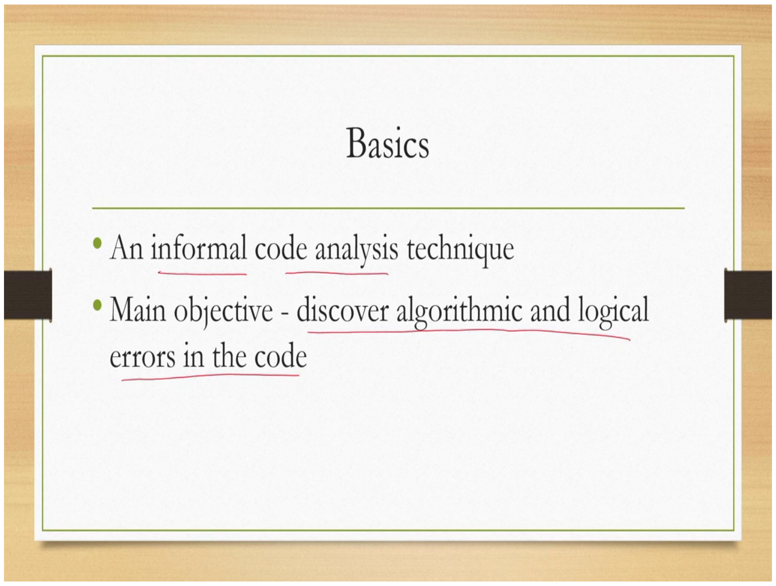


Basics

- An informal code analysis technique
- Main objective - discover algorithmic and logical errors in the code

Here the main objective is to discover algorithmic and logical errors in the code. Remember we mentioned two error types one is logical error other one is violation of conventions or standards in Code walkthrough our primary objective is to identify logical errors rather than the violation of Standards. However while performing walkthrough it is also possible to check for violation of standards that can be one of the minor objectives of cognitive work through.

(Refer Slide Time: 10:42)

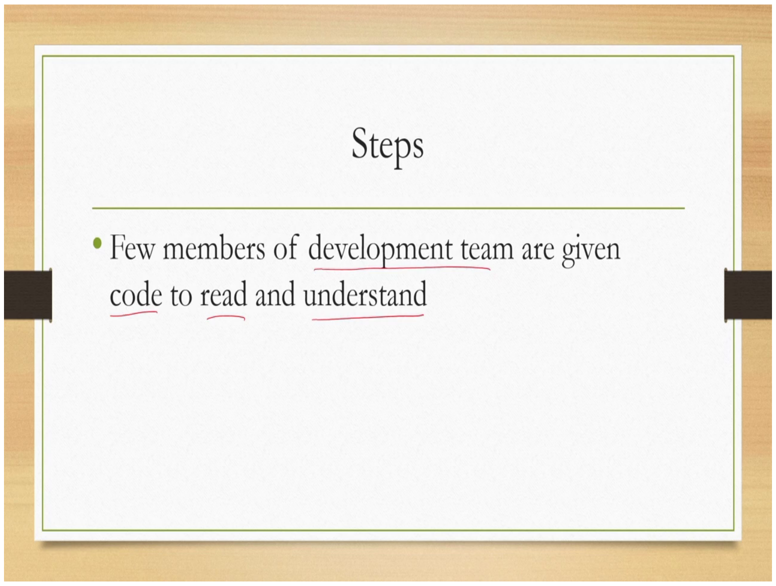


Basics

- An informal code analysis technique
- Main objective - discover algorithmic and logical errors in the code

So, what are the steps for performing code work through. First thing is we have to have a team, team of evaluators. Now the team members who can be the part of the development team each of them is given the code to read and understand. So, we have a team of evaluators who can be part of the development teams and each of them will be given the code and they will be asked to go through the code read it and understand the code.

(Refer Slide Time: 11:28)



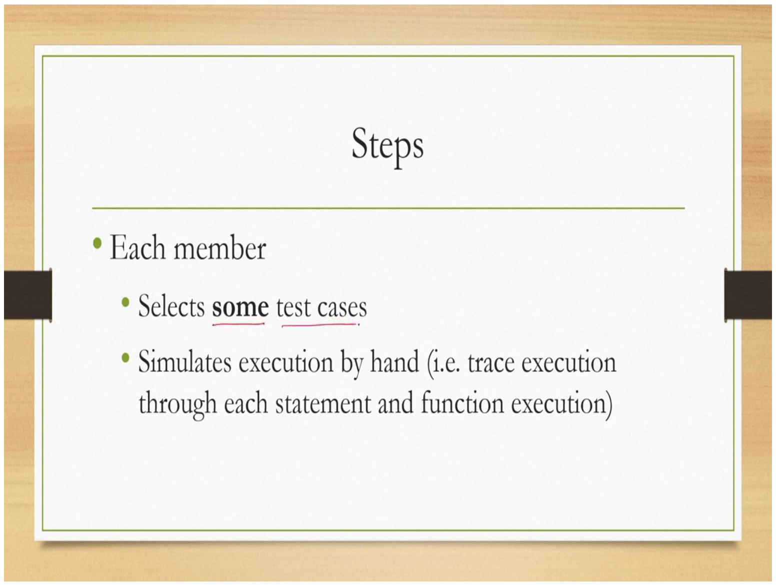
Steps

- Few members of development team are given code to read and understand

Next each of these members is required to select some test cases. Remember we mentioned what is a test case that is an input output combination input is given to the code and output is generated and then we match this with the predetermined output to see if there is a match otherwise there is some logical error. So, each member is asked to select some test cases and then they are asked to simulate execution of the code by hand.

Now this is a term that should not be taken literally by hand means effectively instead of using the computer to execute they have to do so manually like the way we calculate something that means trace execution of the steps in the code through each statement and functional execution. So, given inputs the team members are required to execute the code that means stress the execution of the statements in the code manually and that is what we are calling execution by hand.

(Refer Slide Time: 13:00)



Steps

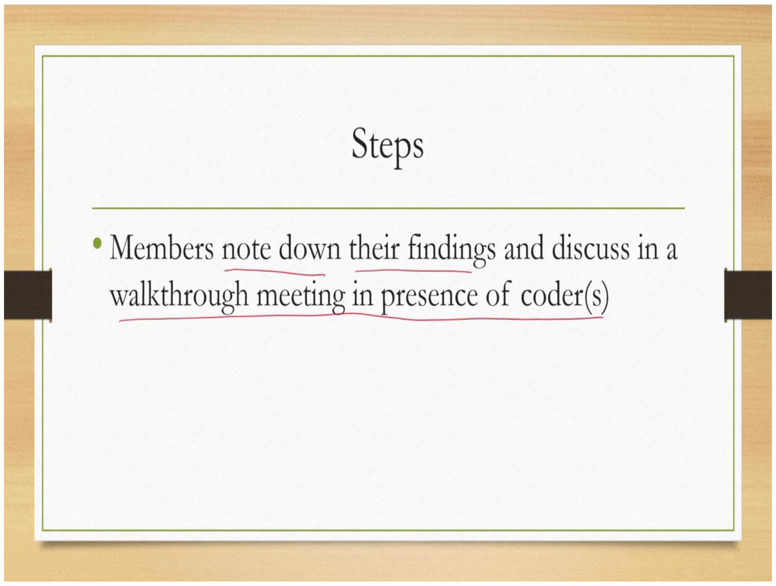
- Each member
  - Selects **some** test cases
  - Simulates execution by hand (i.e. trace execution through each statement and function execution)

After the execution for each test case the members are required to note down their findings and discuss in a walk through meeting in the presence of the coders. So, that is what is required at the end of evaluation by each member. So, to repeat there is a team each team member is given the code they have to read and understand it then each team member is expected to come up with some test cases it is desirable that the test cases are supplied by the coders but then there is a possibility of biasedness.

So, ideally it should have been designed by someone who is not part of the coding team and there should be more than one test cases ideally. For each test cases each member of the evaluation team is expected to hand execute the code and produce output and then match with the desired output and they compile for all the test cases and at the end there will be a meeting where all these findings will be discussed in the presence of the coders.

So, that coders know for which inputs there are some errors and they can accordingly Rectify those parts of the code.

(Refer Slide Time: 14:32)


Steps

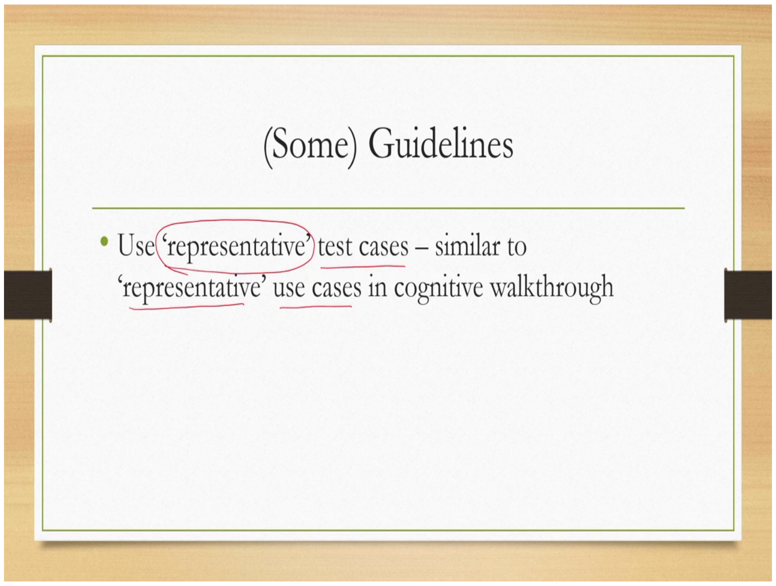
- Members note down their findings and discuss in a walkthrough meeting in presence of coder(s)

While going for code walkthrough some guidelines should be followed first of all the team or the evaluation team that is performing the work through should not be either too big or too small. This is similar to the cognitive walkthrough scenario we discussed earlier. So, ideally three to five members should be there in the team and the team should not be too big or too small. While choosing test cases it is desirable that representative test cases are chosen.

This is similar to the concept of representative use cases in the case of cognitive walkthrough. Now how to determine what is a representative test case again that is not a very easy thing that requires expertise that requires experience. However it is desirable that you come up with representative test cases rather than any random test cases. Remember in our earlier lecture we mentioned that randomly designed test suits not necessarily serve the purpose of identifying errors in a code instead we should be systematically designing test cases.

Since code walkthrough is not a very formal method. So, here of course we can go for informal way of designing test cases however it is desirable to keep in mind while going for test case design that the test cases should be as representative as possible.

(Refer Slide Time: 16:16)

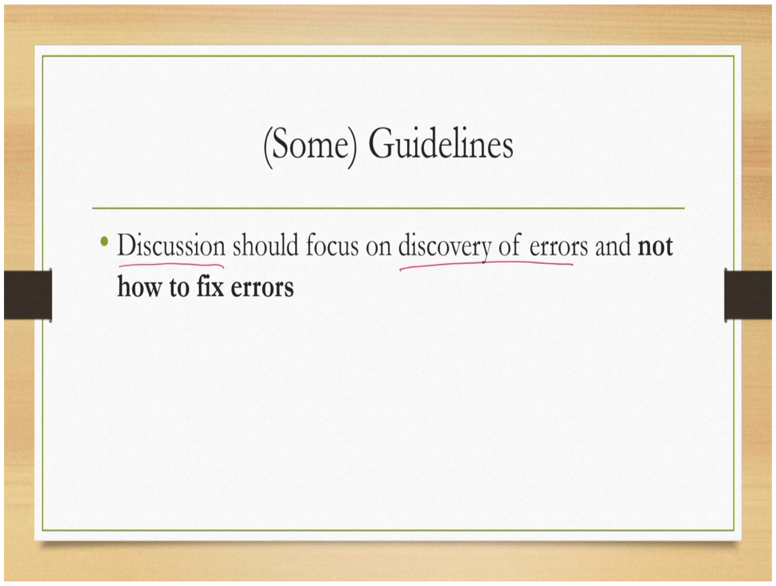


(Some) Guidelines

- Use 'representative' test cases – similar to 'representative' use cases in cognitive walkthrough

Also at the end of this team meeting where findings are discussed in the presence of coders we should keep in mind that during this discussion the focus should be on the discovery of errors and not how to fix errors, fixing errors will come later. During the discussion it should be focused solely on identifying the errors. So, that is in summary what a code walkthrough method the other method for review based testing is code inspection method let us quickly have a look at what this method is and how it can be utilized to test codes.

(Refer Slide Time: 16:56)



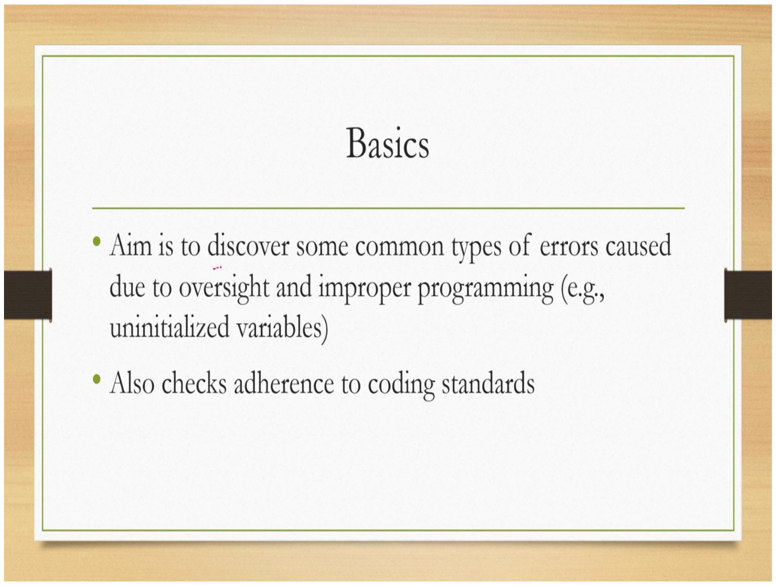
(Some) Guidelines

- Discussion should focus on discovery of errors and **not how to fix errors**

So, here the aim is to discover some common types of errors that are caused due to oversight and improper programming. For example uninitialized variables also this method is more suitable to check for adherence to the coding standards. So, remember in the earlier case that is code walkthrough method our primary objective was to identify logical errors in contrast in the code inspection method primary objective is not to identify logical errors rather identify common mistakes that happen during programming as well as adherence to coding standards whether that is being followed or not.

So, the objectives are somewhat different for these two methods but both provide a quick and cost effective way of identifying or achieving these objectives.

(Refer Slide Time: 18:05)



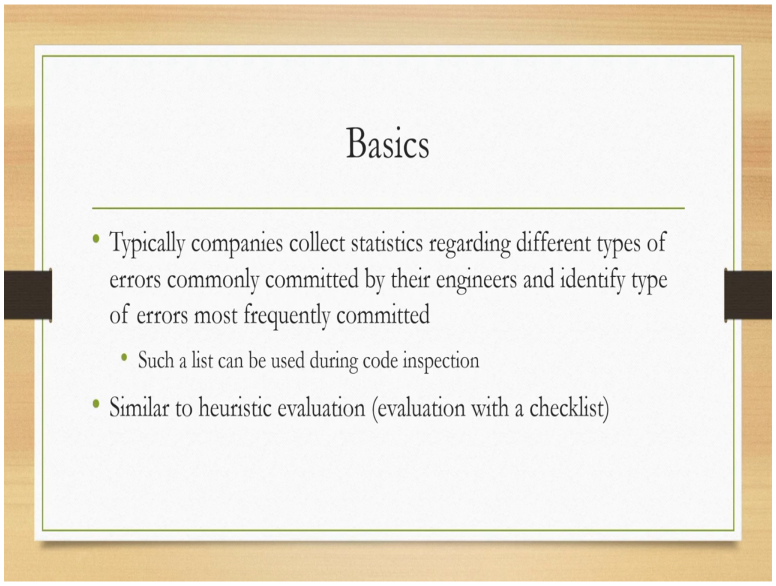
Basics

- Aim is to discover some common types of errors caused due to oversight and improper programming (e.g., uninitialized variables)
- Also checks adherence to coding standards

So, one thing that we should know is that typically companies collect statistics several statistics regarding different types of errors that are commonly committed by their engineers and programmers and identify the type of errors that are most frequently committed. So, that is a common practice across industries that these companies generally collect the statistics regarding different types of errors that happen in the programs written by their engineers and the most frequently occurring errors.

Now that list can serve as a checklist during code inspection. So, in code inspection what is required is a checklist which the evaluators will use and see whether the items in the checklists are being addressed in the code or not. This is conceptually similar to the heuristic evaluation that we have discussed earlier.

(Refer Slide Time: 19:18)



Basics

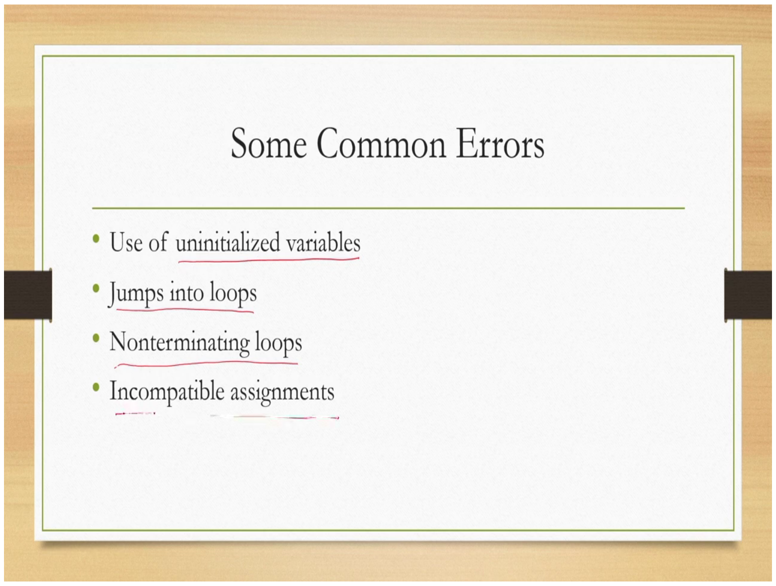
- Typically companies collect statistics regarding different types of errors commonly committed by their engineers and identify type of errors most frequently committed
  - Such a list can be used during code inspection
- Similar to heuristic evaluation (evaluation with a checklist)

So, what are these common errors let us see some examples. One very common error is use of uninitialized variables. Probably many of you have faced this while writing a program we often declare variables but then we forget to initialize it. So, that is one very common type of errors that is found across codes and in a checklist that can be used during code inspection. If we keep such errors in the list then we can check whether in the code that we are evaluating such type of errors are occurring or not.

Another common errors another common error is jumps into Loops this is also quite common and checking against it is required during inspection. Often we forget to terminate a loop in fact often we forget to give proper termination conditions that is again quite common maybe many of you have faced this issue. So, keeping it in the checklist helps while going for code inspection then incompatible assignments.

So, we assign something to something where their types mismatch and that is again quite common many of us faced it while we are writing our programs this is not necessarily due to our ignorance rather many a times we commit this error due to oversight so keeping it in the checklist helps.

(Refer Slide Time: 21:02)

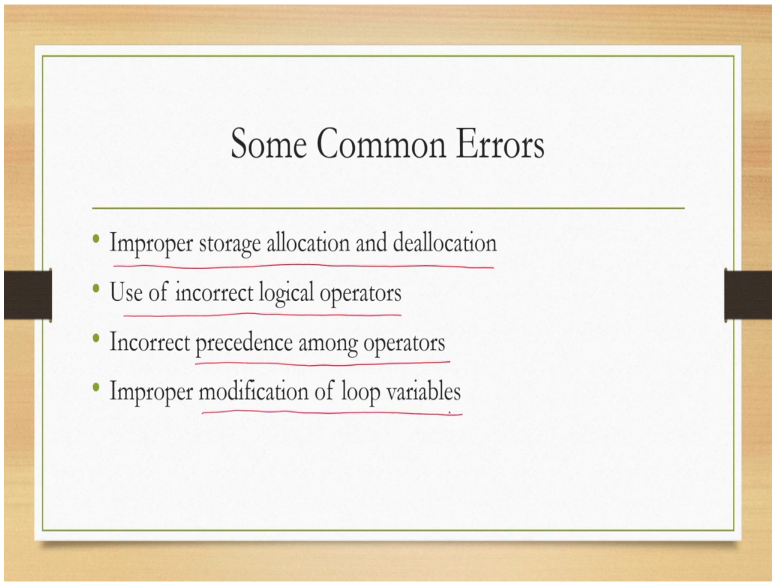


Some Common Errors

- Use of uninitialized variables
- Jumps into loops
- Nonterminating loops
- Incompatible assignments

Improper storage allocation and deallocation is another common errors which we should be aware of file testing a code. Use of incorrect logical operators yet another common error that we should be aware of while testing our code a very common thing is use of incorrect precedence among operators. So, often we do not pay attention to the precedence of operators and we use them in very incorrect way which creates logical errors in the code. So, we should avoid that.

(Refer Slide Time: 21:53)



Some Common Errors

- Improper storage allocation and deallocation
- Use of incorrect logical operators
- Incorrect precedence among operators
- Improper modification of loop variables

So, checking against such type of incorrect usage helps in keeping the code clean improper modification of loop variables another common error we often make this mistake and it is preferable to check against it in your code testing. So, that is in summary what we can do during code inspection. Now few things you should remember one is like code walkthrough code inspection also should be performed by a team again consisting of three to five members.

Each member is given the checklist and each member checks the code against the items in the list and then generates a report again there can be a meeting like code walkthrough at the end of the evaluation in the meeting coders can be present or rather should be present where all these findings are to be discussed. Here again the discussion should focus primarily on identification of Errors rather than how to solve it.

So, all the guidelines that we have covered in code walkthrough applies in code inspection as well. So, just to summarize in review based methods we follow either walk through or inspection method. Each of these methods has its own objective in case of walkthrough objective is to identify most important logical errors. In case of inspection the objective is to identify some common type of errors that may happen as well as whether the code is following coding standards.

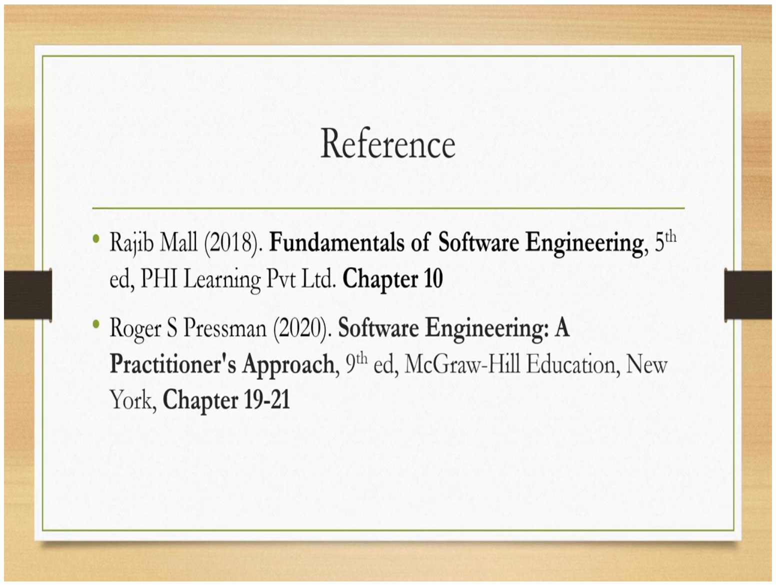
Whether we will be able to identify common logical errors in code walkthrough depends on how the test cases are chosen if they are representative test cases. Then it is more likely that during walkthrough most important logical errors can be found out. Now in walkthrough what happens is that the evaluator executes the code by hand. So, essentially traces the steps or the statements that are part of the code one by one for a given input and at the end calculates the output that should be outputted by the code and matches with the actual output to see if there is any problem or not.

In inspection no execution is required hand execution is required rather the evaluator simply goes through the code and sees whether common errors are present such as uninitialized variables improper assignments improper termination and so on. For these errors to identify these errors no execution is required. Also while going through the code the evaluator look for violation of coding standards such as improper naming conventions and improper function lengths use of too many Global variables.

In a nutshell whatever we discussed in our earlier lectures about good coding practices those things are also observed file a member of the evaluation team goes through the code in code inspection method. In both the cases at the end each member produces a report those reports are compiled and meeting is organized where coders are present. In the meeting all these reports are discussed to identify errors.

Now the focus of the meeting should be to identify errors rather than to take care of the errors that is about code review. Later on we will see one case study on how to create a code review document. I hope you have enjoyed the material you have learned about the basic concepts of code review and you will be able to actually review a code and create a document. In the next lecture we will see how to create such a document.

(Refer Slide Time: 26:14)



Reference

- Rajib Mall (2018). **Fundamentals of Software Engineering**, 5th ed, PHI Learning Pvt Ltd. **Chapter 10**
- Roger S Pressman (2020). **Software Engineering: A Practitioner's Approach**, 9th ed, McGraw-Hill Education, New York, **Chapter 19-21**

The material that I covered today can be found in these books fundamentals of software engineering chapter 10 as well as software engineering a practitioners approach chapters 19 to 21. You will find more details than what I covered but it is always good to go through things in more details that is all for this lecture looking forward to see you all in the next lecture, thank you and goodbye.