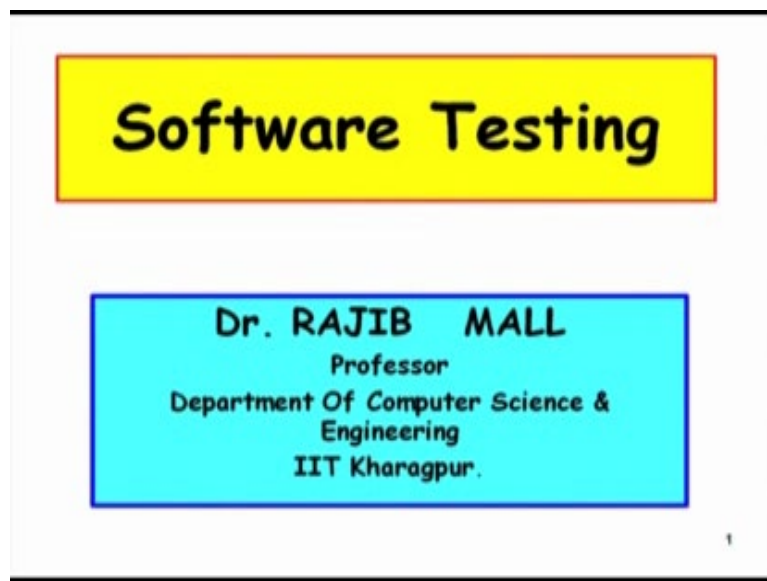**Software Testing**
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 01
Introduction

Welcome to this course on Software Testing.
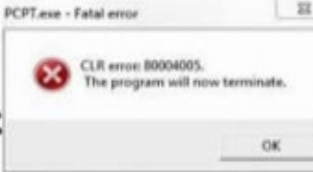
(Refer Slide Time: 00:26)

Over 20 half an hour slots, we will discuss some very basic issues on program testing. We expect that viewer should have at least written some programs and some experience in programming that will help in understanding these lectures.
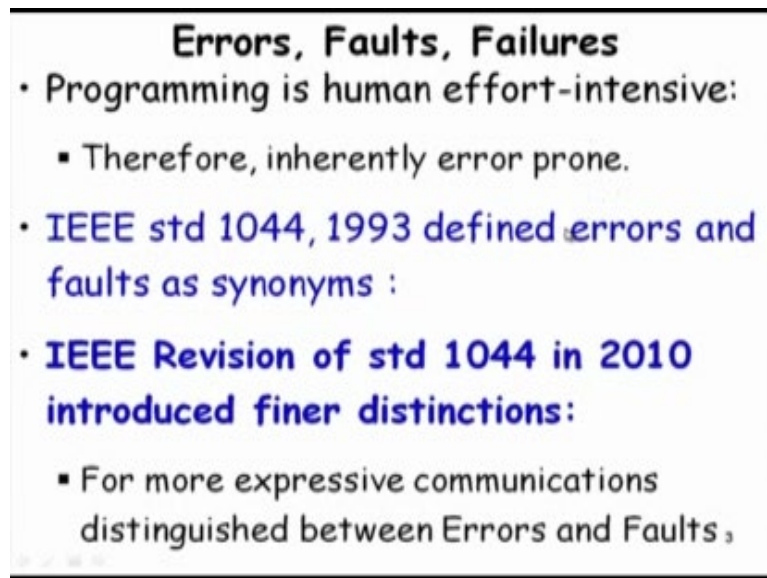
So, now let us get started. When anybody writes a program, there may be lot of errors committed, specially the new programmers, even the professional programmers they commit mistakes. And when there are errors in the code the program might fail during testing. For example, you might see a program crash or wrong results and so on.

Now, let us get our terminology correct. So, when there are faults in the program which we'll also call as defects or bugs a program may fail during testing, but then even if there are bugs or defects or errors in a program those are synonymous terms. Still, a program may not fail during testing, why is that? Several reasons may be the test case did not expose that bug or may be that the bug did not make, it was not easily causing a failure of the program, but then when there a bugs the program is likely to fail. Now let us get these terms correct, the terms because you are going to deal with this terms – Error, Faults and Failures.

Whenever anybody writes program, whether it is a new programmer or a very experienced programmer errors are bound to be there, because these are manual activities, program writing and these are inherently error prone.

These terminologies have been defined in the IEEE standard document 1044 in the document released in ( ) 1993 errors, faults, bugs; all were considered as synonyms. But then there was a revision of the same document released in 2010, the terms needed to be revised because it was considered that having so many terms error, bugs, defects, faults, denoting the same thing is not achieving much purpose, we can have finer meanings conveyed when we use these for different meanings.
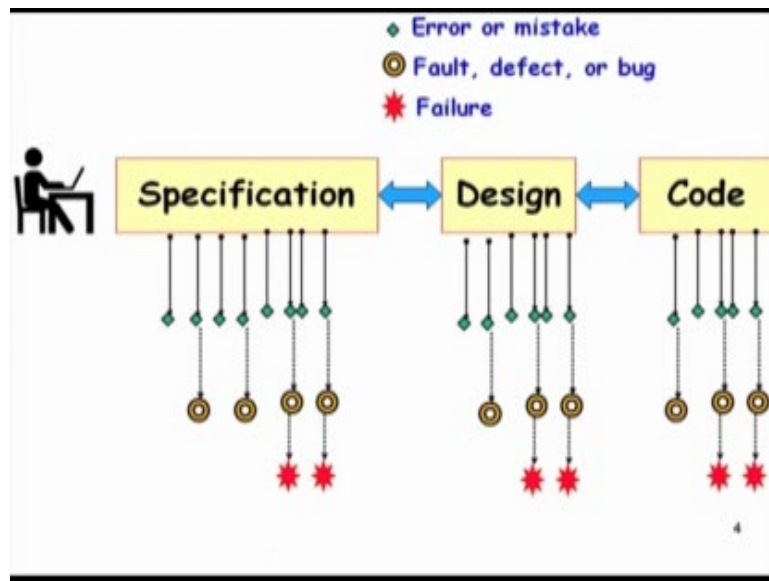
And for this purpose, the IEEE document distinguished between an error or mistake, these are synonyms, error and mistake, fault, defect or bug were also synonyms, but between these there is a difference.

A programmer develops software, by first developing specification, design and code and during each of these activities he may commit an error or mistake. The programmer commits the error or mistake and these we are showing by these green rectangles - green diamonds. But then every error or mistake may not actually cause a fault, defect or bug. Fault, defect and bug, these are synonyms. Every error on the part of the programmer need not cause a fault in the program code.

May be the programmer wanted to write i into j, but then he by mistake wrote i into 2, but then at that point in the code j always has value 2. So, it does not lead to a bug. The program works satisfactorily because j was initialized to 2 and he wrote i into 2 in place of i into j. So, even the program the programmer committed a mistake, but still it did not cause a fault. Now all faults that are there, all faults they may not cause ( ) failures. Some of the faults may be the test cases did not execute, those are very rarely executed once or may be even though there is a fault still it does not cause a program failure, because still possibly the result is acceptable.

(Refer Slide Time: 06:27)

Now, let us see the kind of mistakes programmers commit. Very experienced programmers they also commit errors, research conducted to find out how many errors are committed by programmers, experienced programmers, reveals that on the average very ( ) fine programmers, even when they write code there are 50 bugs per 1000 lines of code and that is quite a significant number and can cause large numbers of failures.

But then, the users do not really get to experience these bugs or failures, because these are tested before releasing. The companies before they release software the test it extensively and results say, that after extensive testing of software still one bug per 1000 lines of code, source code still remain. And these bugs that the programmers commit, what are the sources of the bug? A major source of the bug about 60 percent, about 60 percent from specification and design and about 40 percent are contributed by the code that is the average statistics.
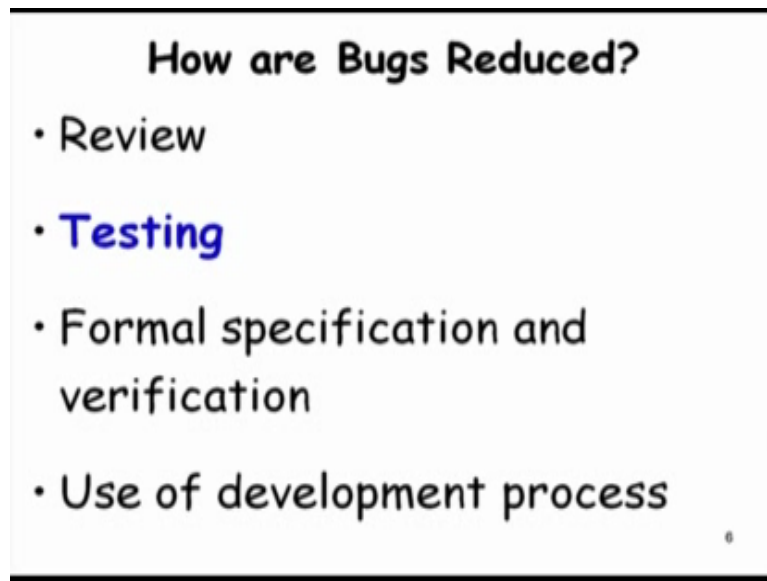
And as we were saying that, a faulty program, a program that fails frequently would not be accepted by ( ) customers, they will reject it and therefore, no company in its right mind will release software with lot of bugs. So, they need to reduce the bugs before they release. How do they reduce the bugs? What are the different techniques that are available to reduce the bugs?

There are 4 primary techniques that companies use, one is review. The faults that are there in the program code are caught during review meeting, even the fault that are there in the specification and design are caught by review meetings and then testing is a very important means of reducing the bugs, formal specification and verification and use of a proper development process. So, systematically developing software with appropriate methodology can reduce the chances of bugs in code ( ).

But before we look at a how do we test a program. Let us see the consequence of not testing adequately a program. There are several incidents that having recorded in the literature, where the cost of not testing has been disastrous. One example that is often sighted is the Ariane 5 rockets, which was launched in early 2000 and it self-destructed just 37 seconds after launch.
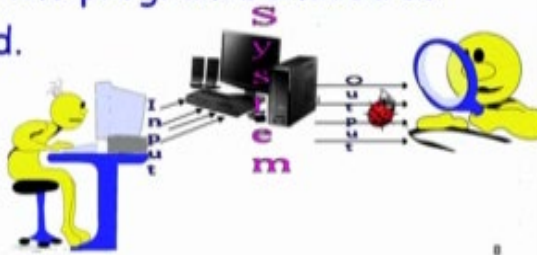
So, that the mission failed, what was the reason? Reason was a bug in the code which was undetected and if you look into details of how the bug was caused you will see that it was due to reuse of earlier code, but then the earlier machine was using a less powerful processor because it was a older machine and then, when it was used for the newer machine the number generated because of the processor's capability, it was a 64 bit processor and then it caused an over flow.

But then it was not detected because just before launch, it was not tested and the exception handler for this was disabled. So, the total cost for this bug is over 1 billion dollars.

So, any software needs adequate testing, but the first question we need to answer is that, how does somebody tests a software? Let us say, let us ask a person who has written a program that how will he go about testing the program. The simplest level he may give an answer that okay, I will input some values, I will observe the result and see if the result is correct or not and then I will keep on inputting values.

So, that is the simplest type of testing where the programmer inputs numbers or inputs data and then observes output to check if the program behaves as expected and if the program output does not match his expectation then, thinks that the program has failed.

Examine Test Result...

· If the program does not behave as expected:

- Note the conditions under which it failed (Test report).
- Later debug and correct.

And, if he finds that the program has failed, prepares a test report. The test report, he would mention that under what conditions it failed, what was the value given and what was the result observed and these test report for all bugs, would be taken up by the development team for later debugging that is locating the source of the error and then correcting the code or may be the specification or designed as the case may be.

Testing Facts

· Consumes the largest effort among all development activities:
- Largest manpower among all roles
- Implies more job opportunities

· About 50% development effort
- But 10% of development time?
- How?

Now, let us see before we start looking into how does one test in more detail, let us see some facts about testing. Among the all the activities that are taken up for developing software, testing actually typically takes the largest effort. All companies spend at least 50 percent effort on testing and 50 percent effort on specifying, designing, coding and so on.

But then one thing that we can say that since a company spends a large amount of effort on testing compared to designing or coding or specifying therefore, the company would need large number of testers and that is true. Just walk into any company and talk to a person and ask what does he do? 50 percent chance, that he would say that I am testing a program, and what really it means that among all the rolls in program development, software development testing has maximum opportunity because most man power is needed on testing.

But then let us just try to understand the situation that companies spend about half the time testing the software, but then when they test, they try hurry up, they try to reduce the time and the 50 percent test effort is spent in 10 percent of the time, development time. So, 90 percent of the development time is taken up in other activities specification, design, coding and that accounts for only 50 percent of the efforts.

The rest of the 50 percent effort on testing is done in 10 percent of the time, how is it possible? It is not hard to guess testing has the maximum parallelism, and that is why number of testers employed by company are larger and they carry out testing in parallel. So, concurrent testing is usually undertaken different parts of the software, different units' integration and even for system testing large numbers of testers can test different aspects of the software; whereas for specification, design, etcetera the number of persons that can work concurrently is restricted, a lot of sequential activities there.

Now, let us look at a few other testing facts the testing over the years has becoming more complex and sophisticated largely because, the programs themselves are becoming large and complex. New programming paradigms have come up for example, web based software or may be embedded software and software running on mobile phones and so on. And also lot of tools has come up.

So, a tester needs to know about these tools and not only that newer testing techniques have been introduced. So, the very ancient in early times when the programmer about 50, 60 years back used to test, he would possibly just do some input, some random values and test called as monkey testing, but then in the last 50, 60 years especially ( ) last decade lot of development has taken place making testing a very specialized and very sophisticated profession. So, let us look at that.

As I was saying that long back, testing was just inputting some values to the program and trying to crash the program or see that the program produces some wrong results. And that is the reason why testing was considered to be not very attractive, did not have too much of intellectual content on that because inputting only random values, but somehow that stigma has remained even though the profession has changed completely.

Now, testing is one of the most challenging works in any company. That over the years testing has taken a centre stage in all types of software development, and no more software is being tested by monkey testing; the monkey testing is just inputting random values and seeing that whether the software is working or not. Now the testers need to have good knowledge of different testing techniques and there are large number of them as we will see in our subsequent sessions, and also the automated tools that have become available must develop a proficiency with those.
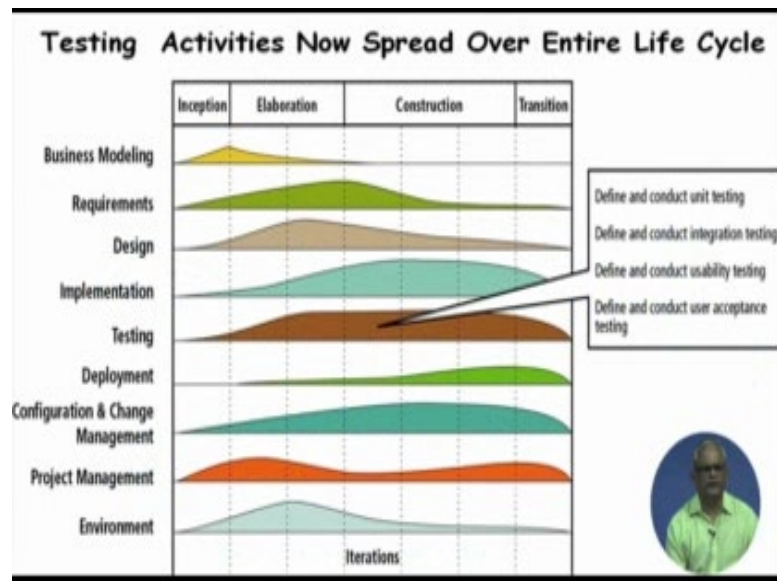
Now, let us answer this question that when is testing carried out in the software development. Okay, if we are considering water fall model of development then testing typically occurs towards the end of the development cycle. So, that is during the coding phase, unit testing is carried out and then integration and system testing is carried out the testing phase. So, using water fall model what do the testers do rest of the time they actually need testers only towards the end in water fall model because we said that there will be large number of testers and then what do they do in the early part of the development cycle in waterfall model.

But in the latter development methodology, software development methodology for example, the unified process or the agile methods, testing is spread all over the development cycle and even the older v model of development has testing spread all over the life cycle. So, the testers are busy doing something or other as defined by the life cycle model and also the software is the bugs are exposed earlier.

Let us see here, this is the effort required in different phases of the unified process. In unified process there are 4 phases inception, elaboration, construction and transition; and just see that the testing effort is there all through. So, what do the testers do? All over the life cycle they define and conduct unit testing, define and conduct integration testing, define and conduct usability testing, define and conduct user acceptance testing.

Actually any iterative development which are the development processes now very popular, the iterative development processes in every iteration is like a small water fall where specification, design, coding and testing is carried out. So, in any iterative development methodology testing is present all through the life cycle and in the water fall and its related development processes, testing is only towards the end.

(Refer Slide Time: 22:45)
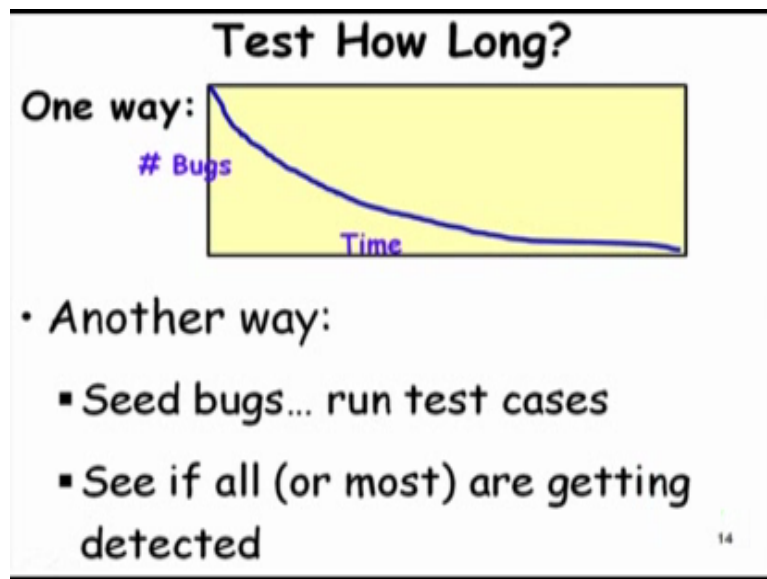
But now let us ask this question that once program is being tested, bugs are detected as you input or give or use more and more test cases more and more bugs are detected, but then the number of bugs being exposed over time falls. So, how do you know, I mean when do you stop testing? One answer is that as the number of bugs reduces if bugs are not found in a day or two of testing. So, may be it is the time to stop testing of course, depends on the specific application for which the software is being developed; but are there other ways of knowing how long to test?

The other thing we can do, the program manager or the leader, he seeds ( ) bugs in the program code unknown to the testers and then the testers test the software, report failures and then these are debugged by the developers and the program manager finds out if all the bugs are, have been detected. If all the bugs that he has seeded or most of the bugs that he has seeded have been detected then he knows that the other bugs that were there in the code are possibly all been detected and that is the time to stop.

So, seeding bugs and seeing how many of them are getting exposed indicates to what extent the software has been tested.

Now, let us get another terminology correct. This is also very frequently use terminology - Verification versus Validation. So, verification is the term that is used to check that whether a work product, as it gets developed in the life cycle whether once at one stage the war product conforms to the work product in the next stage. So, whether the output of one phase that is the work product produced after one phase conforms to it is a previous phase, we verify whether a war product conforms to its previous phase, how does one verify?

Verification is checking whether a work product conforms to its previous phase work product, but how does one verify? Verification techniques are review simulation etcetera, but in contrast to verification, validation is the process of determining whether the fully working software conforms to the requirement document.

(Refer Slide Time: 26:16)
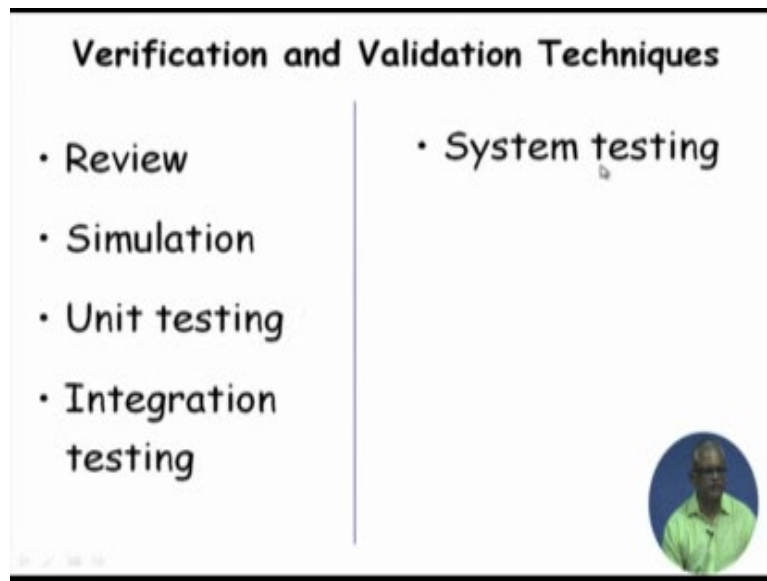
So, we can say that verification is concerned with phase containment of errors that is as the errors are getting detected during the development cycles, during the development cycle whether it is getting detected and eliminated. Whereas validation is concerned with ensuring that the product is error free. So, in other words we can say that verification is concerned with checking whether we are developing it right and validation is concerned with checking whether we have developed the product right. So, whether the product is right, whereas in verification we check whether we are developing it in a right way whether or the process or the work product is being developed correctly.

So, these are some of the techniques for verification and validation, verification we use review, simulation, unit testing, integration testing and system testing. Does it appear surprising that we are written unit testing and integration testing are verification techniques? Yes, these are verification technique because unit testing is used to check whether a unit conforms to its design. Unit may be function or a module and similarly integration testing whereas system testing is a validation technique.

We will stop at this point and continue from this point in the next session.

Thank you.