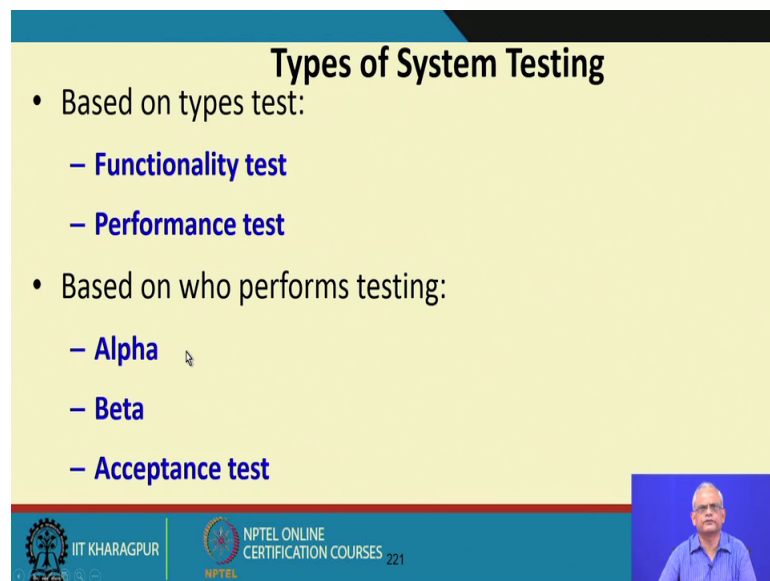


Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 44
Basic concepts in Testing-II

Welcome to this lecture. In the last lecture, we had started discussing the testing issues looked at some very basic concepts. Today let us build on the aspects that we had discussed last time. If you remember in the last lecture, we had discussed about the types of testing. Unit, integration, system these are the 3 levels of testing; unit, integration and system. And then we have the regression testing, which is a different dimension of testing. We had looked at some very basic concepts of unit testing and integration testing in the system testing we just started to discuss last time.

(Refer Slide Time: 01:15)



Types of System Testing

- Based on types test:
 - **Functionality test**
 - **Performance test**
- Based on who performs testing:
 - **Alpha**
 - **Beta**
 - **Acceptance test**

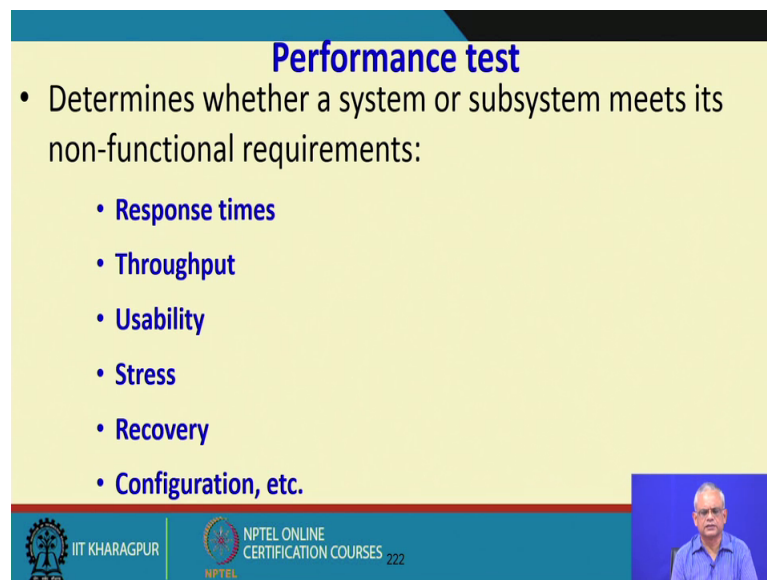
The slide also features logos for IIT Kharagpur and NPTEL Online Certification Courses 221 at the bottom, and a small video inset of Prof. Rajib Mall in the bottom right corner.

Now, let us proceed further. We can consider system testing to be 2 types, either a functionality test or performance test. In other words, if we have to do a system testing for some software, we have to write 2 types of test cases. One is the functionality test cases and the other is the performance test cases. We will see what the functionality tests need to test which aspects and which aspects will be tested by the performance test.

We can also consider the system testing to be of 3 types. Based on who performs testing, there are 3 different types of system testing. The alpha testing is done by the developing

the developing organization the testers in the developing organization. This is a type of system testing, which is first performed and after the system passes the alpha testing, beta testing is performed; beta testing is performed by a set of friendly customers or users.

(Refer Slide Time: 02:59)



Performance test

- Determines whether a system or subsystem meets its non-functional requirements:
 - Response times
 - Throughput
 - Usability
 - Stress
 - Recovery
 - Configuration, etc.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 222

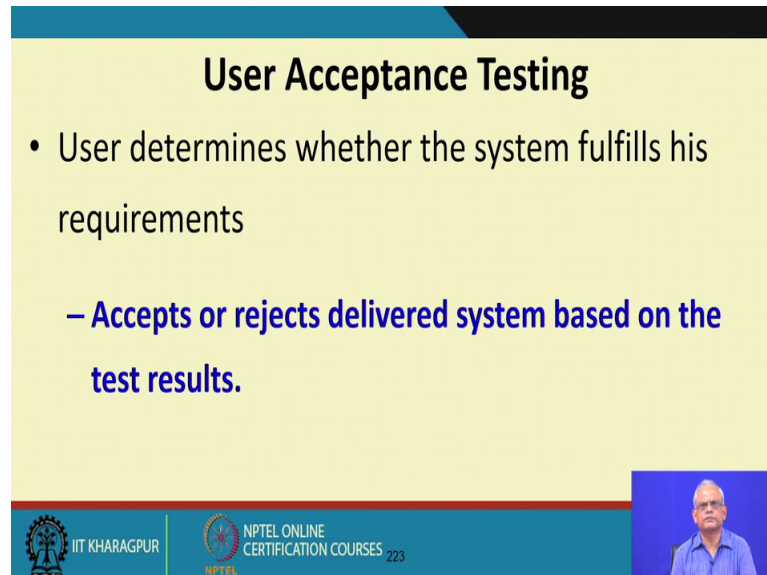
And after all the issues that arise during beta testing are address the acceptance testing is performed and acceptance testing is performed by the customer. Once the system is delivered to the customer perform the acceptance testing. And, they accept the software if it passes the acceptance this thing otherwise, they return it they do not accept it.

Now, as we proceed we will look at the functionality test. The functionality test is basically all the functions, that are documented in the SRS document, this tested whether all those functions are working fine. The second category of tests that have to be developed the test cases and to be executed are the performance tests. The performance tests determine, whether a system or subsystem satisfies the non-functional requirements.

So, both the functionality and the performance test in system testing are designed based on the SRS document. The functionality tests are designed based on the functional requirements and the performance tests are designed based on the non-functional requirements. There are several types of functional tests. The response times, whether all the response times are met throughput, usability, stress, recovery, configuration, etcetera. There are many types of performance tests. And, later this lecture series we will see what

are the aspects that are tested in this different types of performance tests throughput usability, stress, recovery, configuration etcetera.

(Refer Slide Time: 05:11)



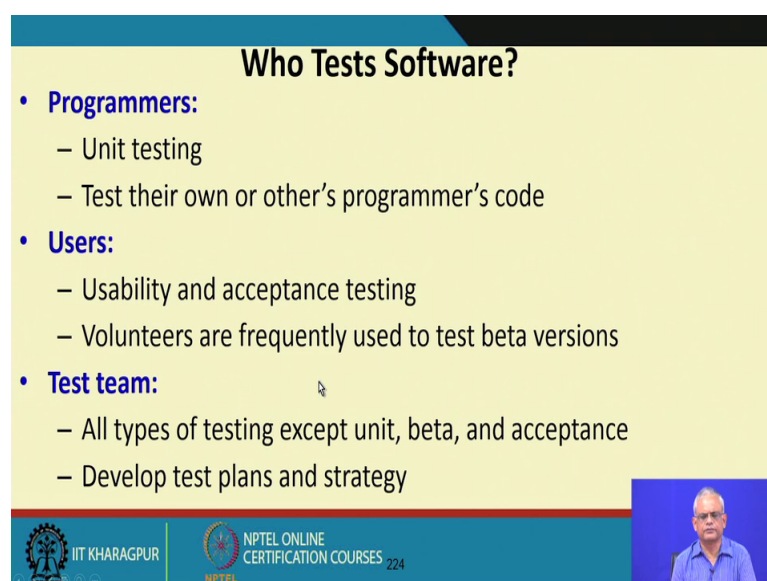
User Acceptance Testing

- User determines whether the system fulfills his requirements
 - **Accepts or rejects delivered system based on the test results.**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 223

Just to have some more basic ideas and testing. Once the software is delivered to the customer, the customer performs what is known as the user acceptance testing or acceptance testing. They just check whether the system fulfils all their requirements and based on their test they either accept or reject the software.

(Refer Slide Time: 05:44)



Who Tests Software?

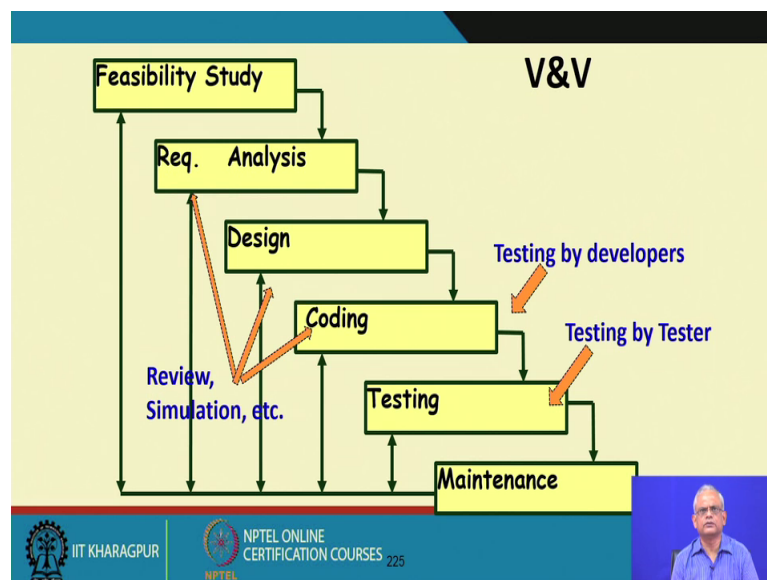
- **Programmers:**
 - Unit testing
 - Test their own or other's programmer's code
- **Users:**
 - Usability and acceptance testing
 - Volunteers are frequently used to test beta versions
- **Test team:**
 - All types of testing except unit, beta, and acceptance
 - Develop test plans and strategy

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 224

Now, based on our discussion so far we can see that a software is tested by different types of personally. One is the developing in the developing organization, the programmers who wrote the software, they perform the unit testing. And, they may also test the other programmer's code if necessary. So, the programmers themselves are one type of tester. The users they perform the usability and the acceptance testing and some of the friendly users they may do the beta testing. And, then often the organizations have a separate testing and they do all testing that is integration alpha testing and so on.

So, they do most of the testing accepting the unit beta and the acceptance testing. And, also they develop the test strategy that what kind of tests need to be performed, and they also develop the test plan when in what sequence the test need to be applied?

(Refer Slide Time: 07:12)



If, we look at the broader picture, in the development lifecycle whether shown a iterative waterfall model. In the initial stages of the waterfall model, you can see that the review is done after the end of every stage. The SRS document is reviewed during the requirement analysis and specification during design, design document is reviewed the code is reviewed. And, there may be necessary to simulate this and these are all done by the developers.

So, in the initial stages the testing is done by developers, but during the testing stage where the integration and system testing is carried out the testing is done by the tester.

(Refer Slide Time: 08:14)

Pesticide Effect

- Errors that escape a fault detection technique:
 - Can not be detected by further applications of that technique.

The diagram illustrates the pesticide effect using four vertical bars labeled 'FILTER' in different colors: green, blue, red, and blue. Purple and blue dots are shown being filtered out by the first two filters, while pink and purple dots are filtered out by the last two. Green and pink arrows point upwards from the bottom, representing pests surviving and multiplying.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 226

Let us look at another very important fundamental concept and testing. And, this concept goes by the name pesticide effect. This concept is given the name pesticide effect; because of it is analogy to the use of pesticide in a crop field. Let us assume that we have a cotton crop and then the farmer found that it is infested with pests. So, what does the farmer do? So, these are the pests who have infested the crop and the farmer would apply some pesticide and then most of the pests get killed, but then few of them are survived.

Now, in the next season, when the farmer plants the crop, again those which are survived they multiply and also new types of pests come. Again the farmer applies pesticide, but then if the farmer applies the same pesticide let us say DDT, those which have survived DDT, again applying DDT will do nothing to them, because, they have their resistant to DDT.

So, the farmer needs to apply a new type of pesticide; let say malathion and then many of the pests get killed, but then those with survived again they appear in the next crop multiply and appear and new types of pests appear. But again applying malathion, we will do nothing to this pests, because they had survived malathion and malathion does not kill them. So, the farmer needs to apply some other pesticide.

In testing it is the same thing, there is a very strong analogy with the pesticide effect in crops. We can state by saying that errors that escape a fault detection technique cannot be detected by further applications of that technique. What it means is that? We have many

types of testing technique, dozens of testing technique we will see some of those may be we will look at a dozen, but then there are several dozens of testing techniques. Each time we apply a testing technique, some of that faults are detected, but then some of the faults escape and if we again and again apply the same testing technique. We will be not detecting those which had already escaped, because this testing technique does not detect those kind of faults.

We need to apply a new type of testing technique or a different type of testing technique and we catch some of the faults and again apply another testing technique. So, the bugs which escape a testing technique, further applications of that same testing technique we will yield very little result. And, that is the reason why? In a software development organisation, several types of testing techniques are used one after other and this is called as the pesticide effect.

(Refer Slide Time: 12:46)

Capers Jones Rule of Thumb

- Each of software review, inspection, and test step will find 30% of the bugs present.

In IEEE Computer, 1996

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 227

Capers Jones who is a well-known researcher in various testing, he had written a article in the IEEE computer in 1996. And, he had proposed a rule of thumb in that article it states each of software radio inspection and tested will find 30 percent of the bugs present. So, what he says is that each of this can be considered as a bug filter. In review, we get some bugs exposed. During inspection, we will get some more bugs exposed and then there are various types of testing and in each of those we get only 30 percent of the bugs present.

So, initially if the code had 1000 bugs review will catch 300 of them, out of the remaining 700 30 percent is 210, 210 will be obtained by inspection and then we have 490. And out of that 490 apply unit testing we will get 30 percent of that, which is about 100 82 or something and so on. So, that is the one he had experimentally observed, that each of the software inspection and testing technique can be considered as a bug filter. And this bug filter can find 30 percent of the bugs present.

(Refer Slide Time: 15:09)

Pesticide Effect

- Assume to start with 1000 bugs
- We use 4 fault detection techniques :
 - Each detects only 70% bugs existing at that time
 - How many bugs would remain at end?
 - $1000 * (0.3)^4 = 81$ bugs

The diagram illustrates the 'Pesticide Effect' where a large group of bugs (represented by blue dots) is filtered through four sequential barriers (represented by blue vertical bars). After each barrier, only 30% of the bugs remain, resulting in a significantly smaller group of bugs (represented by blue dots) after the fourth barrier.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 228 | NPTEL

Now, let us do a small arithmetic small problem based on this pesticide effect. Assign that we had thousand bugs. And, we deployed 4 fault detection techniques and each technique is highly successful unlike, what Capers Jones had written our techniques could detect 70 percent of the bugs existing at that time.

So, after the detect application of 4 detection techniques, how many bugs should remain at the end? So, we can do it initially after the first technique 300 will survive 70 percent will get detected out of 1000. So, that is 700 detected in 300 existing and then again 30 percent of that will survive and so on. We can do it quickly when noting that it is equal to 1000 into 0.3 to the power 4, because each time only 0.3 survive.

So, the total surviving bug out of the application of these 4 techniques is 81 bugs in this program. And this is a big number of bugs 81 bugs, when the user's encounter this bugs in failure. They would be extremely unhappy and they will call it as a poor quality

software. And therefore, we need to have many more bug filters such that the bugs reduce to very small level.

(Refer Slide Time: 17:05)

The slide features a diagram of the iterative waterfall model on the left and three quiz questions on the right. The diagram consists of six yellow boxes arranged in a descending staircase pattern from top-left to bottom-right. The boxes are labeled: Feasibility Study, Req. Analysis, Design, Coding, Testing, and Maintenance. Arrows point downwards from each box to the next one below it. Additionally, vertical arrows point upwards from the bottom of each box to the top of the box immediately above it, representing feedback loops between stages. The quiz questions are:

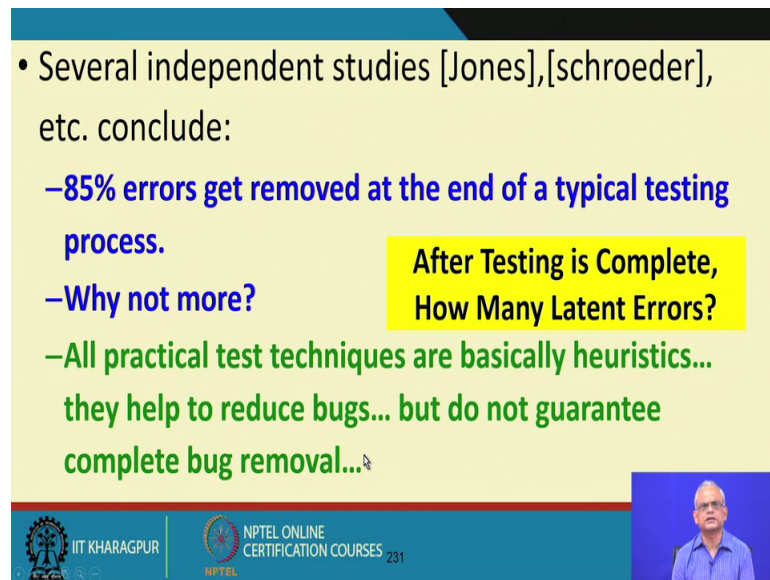
1. When are verification undertaken in waterfall model?
2. When is testing undertaken in waterfall model?
3. When is validation undertaken in waterfall model?

The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES 229 logo, and a small video inset of a man in a blue shirt.

Now, let us have a small quiz given that this is the iterative waterfall model, when exactly are the verification tasks undertaken in this model. In which stages are the verification work undertaken ok. The verification the form of review inspection simulation etcetera are done on during requirements analysis design and coding that is answer, when is testing undertaken in waterfall model ok, to answer this you can say that the unit testing is done during the coding stage.

And, the integration and system testing is done during the testing stage, when is the validation undertaken in waterfall model. Validation, if we remember from our last lecture discussions, validation is basically system testing and system testing is undertaken during the testing stage. Now, let us look at some more basic concepts in testing.

(Refer Slide Time: 18:33)



- Several independent studies [Jones],[schroeder], etc. conclude:
 - 85% errors get removed at the end of a typical testing process.
 - Why not more?
 - All practical test techniques are basically heuristics... they help to reduce bugs... but do not guarantee complete bug removal...

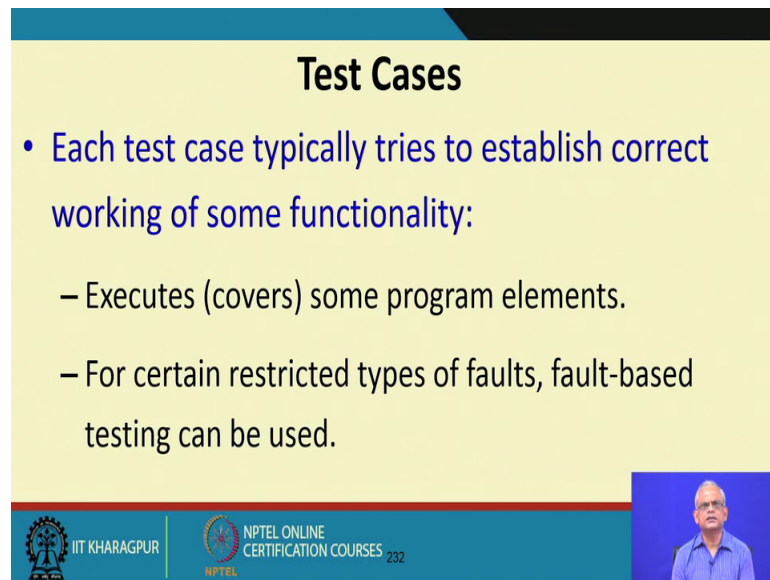
After Testing is Complete,
How Many Latent Errors?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 231

Let us assume that you have tested one software very thoroughly or let us say developing organization, developed a large software and then tested it very thoroughly using the available techniques. How many errors will survive, after all the testing activities are over and success report for all the test have been obtained. Several studies have been undertaken on this issues Jones, Schroeder etcetera, but then they conclude that a typical testing process removes 80 percent 85 percent of the error and 15 percent continue to stay and those are called as a latent errors.

But, then why cannot we remove more errors, can not the testing techniques remove 99.9 or 100 percent of the errors, why they remove only 85 percent of the errors. The answer to this question is that all the testing techniques that we will discuss are basically heuristic technique. And, these do not guarantee that they will detect all errors. So, all the testing technique are heuristics they help to reduce the bugs, but provide no guarantee of complete removal of bugs this is a important observation, that all testing techniques which you discuss are heuristics. And, they help to only reduce bugs do not provide any guarantee concerning complete bug removal.

(Refer Slide Time: 20:40)




Test Cases

- Each test case typically tries to establish correct working of some functionality:
 - Executes (covers) some program elements.
 - For certain restricted types of faults, fault-based testing can be used.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 232

NPTEL



What do you mean by test cases? A test case we execute to establish their some functionalities working correctly. When, we execute a test case some of the program elements are executed. And therefore, we can consider the effectiveness of a test case based on how much coverage is achieved by that test case? So, the coverage measurement of test cases is a important metric for test cases, but then we also have another one, many test cases we design them or detecting specific types of faults.

For example, arithmetic faults and so on. So, we call them as fault based test cases. So, the test cases are 2 types; one we measure the effectiveness of the test cases by the coverage achieved and the other is how many faults they are targeting.

(Refer Slide Time: 22:08)

• Test data: **Test data versus test cases**

- Inputs used to test the system

• Test cases:

- Inputs to test the system,
- State of the software, and
- The predicted outputs from the inputs

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 233

Now, let us look at a finer distinction between a test data and test case. Test data as the name says it just the inputs test inputs that are given to the system, these are the data basic data to execute. Whereas, the test cases not only contain the input data to the system, not only describe the input data to be given to the system, they also describe at which state of the software this data needs to be input. And also the predicted output that should be obtained from the input.

So, test data is just a component of the test cases and in addition the test cases contain the state at which the input is to be given. And also what is the predicted output corresponding to the inputs.

(Refer Slide Time: 23:13)

Test Cases and Test Suites

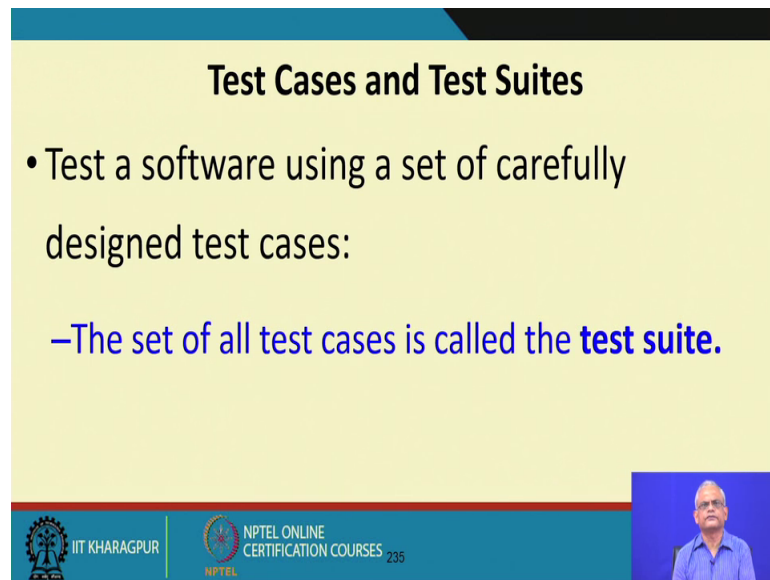
- A **test case** is a triplet [I,S,O]
 - I is the data to be input to the system,
 - S is the state of the system at which the data will be input,
 - O is the expected output of the system.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 234

Based on that we can say that a test case has 3 components I, S and O, I is the input data or test data to the system, S is the state of the system at which the data will be input, and O is the output that should be expected. Just to give an example of the state at which the data to be input, let us consider a functionality in a library system; like let say renew book.

For a renew book test case to be executed, we must have the same book already issued out to member, the book should have been created that should exist in the library and then it must have been issued to a member and only then that member can renew it. So, we call that as the state of the system that is the, unless the book has been created. And the book has been issued out this test case of renew book will not work.

(Refer Slide Time: 24:40)



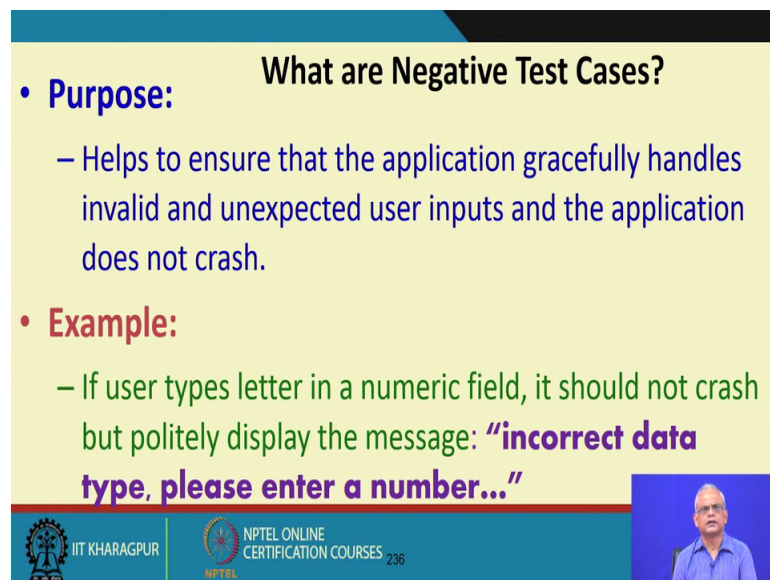
Test Cases and Test Suites

- Test a software using a set of carefully designed test cases:
 - The set of all test cases is called the **test suite**.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 235

Every, software is the tested using a set of test cases, which are carefully designed and all this test cases, which have been designed to test the software is called as a test suite.

(Refer Slide Time: 24:53)



What are Negative Test Cases?

- **Purpose:**
 - Helps to ensure that the application gracefully handles invalid and unexpected user inputs and the application does not crash.
- **Example:**
 - If user types letter in a numeric field, it should not crash but politely display the message: **"incorrect data type, please enter a number..."**

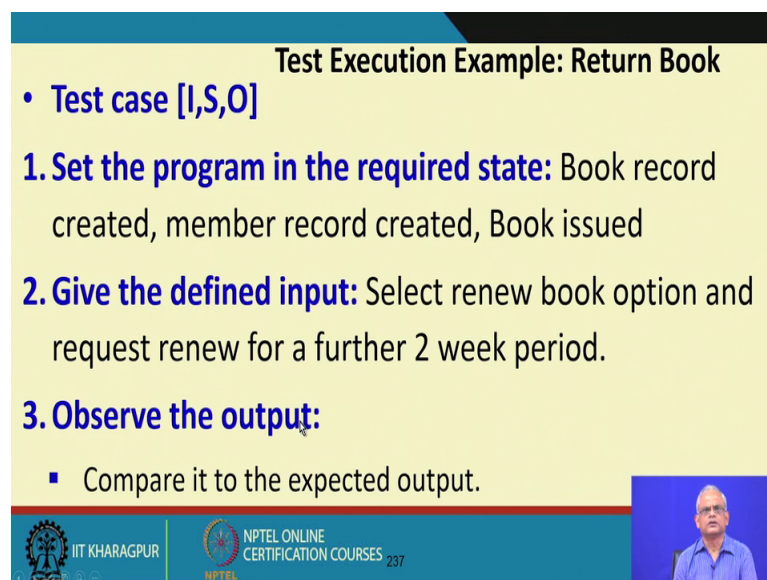
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 236

Now, let us see another basic concept, which is about negative test cases we have been. So, far discussing about positive test cases which detect bugs in the code, that is the software does not do something it should have been doing, but negative test cases this the provide unexpected are invalid inputs. And then see how the system behaves? The system should behave gracefully, it should not crash.

For example, if it wants the number of books to be entered number of books to be issued is to be entered, and the librarian entered let say instead of a number just entered something like a digit sorry an alphabet like a or something by mistake, and then the system should not crash.

So, the a m of the negative test case is to give some input, which is not really correct input to be given as per the SRS document these are wrong inputs, but just to check that the system behaves acceptably and does not crash. For example, in a numeric field a user types a alphabet, and then it is to be observed that the system does not crash and displays a polite measure, that data type entered is incorrect please enter a number.

(Refer Slide Time: 26:57)



Test Execution Example: Return Book

- **Test case [I,S,O]**
 - 1. Set the program in the required state:** Book record created, member record created, Book issued
 - 2. Give the defined input:** Select renew book option and request renew for a further 2 week period.
 - 3. Observe the output:**
 - Compare it to the expected output.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 237

But, as the test execution is done the test cases have been designed, how is the test execution done? All the test cases have been designed and now the test execution is to be done test have to be executed. So, the program has to be put in the required state, the input has to be given and the output has to be observed. We are almost at the end of this lecture and with this set of basic concepts on testing, we just stop here and we will continue in the next lecture.