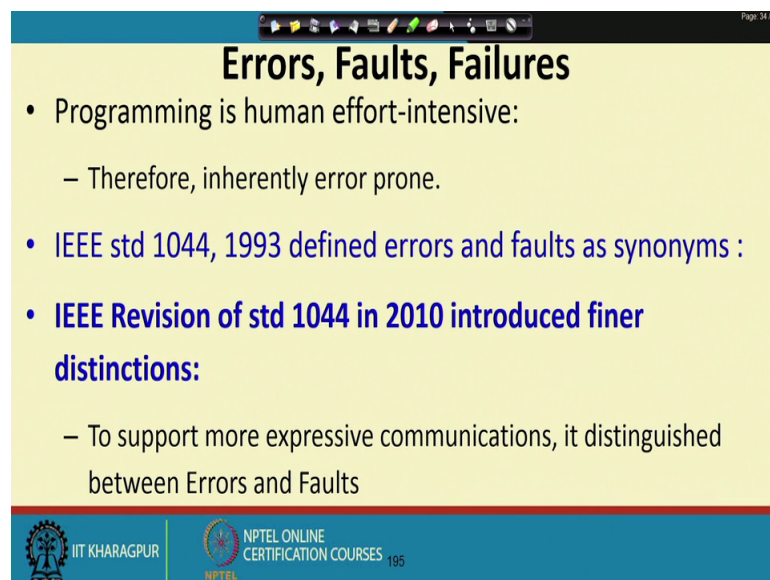


Lecture – 43
Basic concepts in Testing-I

Welcome to this lecture. In the last lecture, we started looking at the testing issues. We are trying to identify some very Basics issues in Testing. And, we had said that, when testing we observe the failures. We do not observe the bugs; we just observe a manifestation of the bug that is failure. And, we are trying to understand the terms error fault and failures failure is caused by faults.



(Refer Slide Time: 00:57)



Page 34 / 34

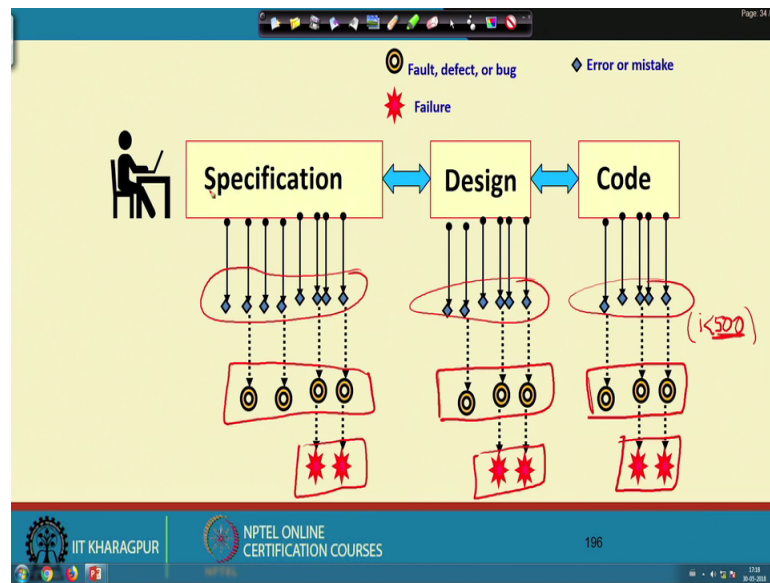
Errors, Faults, Failures

- Programming is human effort-intensive:
 - Therefore, inherently error prone.
- IEEE std 1044, 1993 defined errors and faults as synonyms :
- **IEEE Revision of std 1044 in 2010 introduced finer distinctions:**
 - To support more expressive communications, it distinguished between Errors and Faults

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES 195

And, we are saying that initially errors and faults bear synonyms, but then later the IEEE standard said that there is a difference between error and faults, because that will help us to express the ideas better. The faults or bugs these are caused by mistakes or errors on the part of the programmer. And, these create faults bugs or defects and this in turn may cause failures.

(Refer Slide Time: 01:41)



So, I represented that in this diagram, the programmer here manually does many activity, does specification, design, code, and the programmer can commit mistakes or errors, can commit many mistakes or errors during specification design code, but not all mistakes become faults or defects. For example, let say the programmer was writing a code i less than 50 and by mistake or the error wrote i less than 500, but then in the program there is no way that I can get a value more than 50. And therefore, even though he made a mistake or a error this does not become a fault or a bug.

So, only some of the errors they become faults or defects or bugs, but then some all the bugs, they do not result in failures. May be the test data that is normally given they do not cause this bugs to express and some of the bugs they cause failures. So, can say that, the mistakes or the errors on the part of the programmer, some of them they cause bugs or faults and some of the faults, they cause failures.

(Refer Slide Time: 03:59)

A Few Error Facts

- Even experienced programmers make many errors:
 - Avg. 50 bugs per 1000 lines of source code
- Extensively tested software contains:
 - About 1 bug per 1000 lines of source code.
- Bug distribution:
 - 60% spec/design, 40% implementation.

Bug Source

■ Spec and Design
■ Code

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us look at some facts about errors. Even the most experienced programmers they do mistakes or errors. And, a typical industry average is 50 bugs per 1000 lines of source code is what good programmers make. And, testing reduces the number of bugs, after thorough testing about a 1 bug per 1000 lines of source code still remain. Even, if we have tested a program very well, still there may be 1 bug for thousand lines of source code, but what are the origin of all these different bugs 60 percent can be trace to specification and design and about 40 percent from code is the average figure ok.

(Refer Slide Time: 05:07)

I FOUND THE ROOT CAUSE OF OUR PROBLEMS.

PROBLEMS

IT'S PEOPLE.

THEY'RE BUGGY.

DID YOU BRING A PEN?

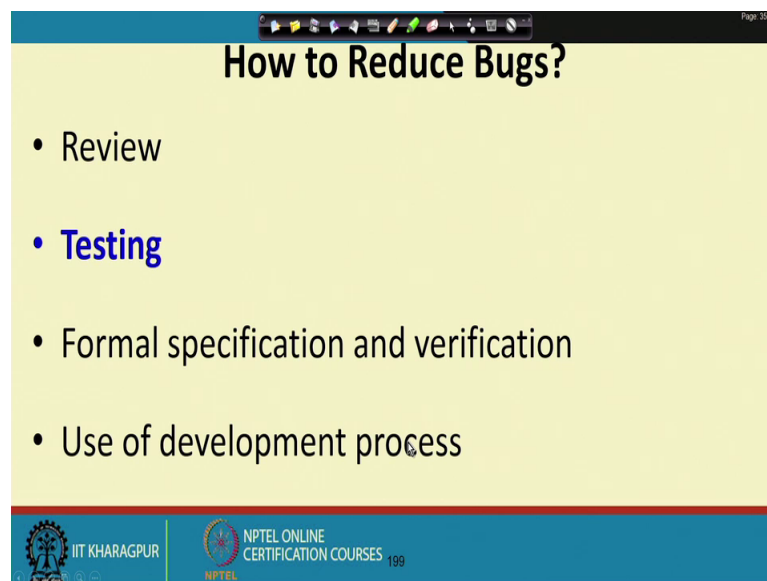
DilbertCartoonist@gmail.com

© 2015 Scott Adams, Inc. /Dist. by Universal Uclick

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 198

This is just a lighter moment a cartoon, but then that holds a message here this is a Dilbert cartoon by from Dilbert dot com. So, this is a engineer he is Dilbert he is giving a presentation says I found the root cause of our problems says, it is a people. They may mistakes; they are buggy and just see here the manager says did you bring a pen. So, he has already forgotten to bring a pen. So, he is made a mistake. So, people forget and they make mistakes, even very experienced people they do mistakes and that is the reason why there are bugs in the code?

(Refer Slide Time: 06:14)



The slide is titled "How to Reduce Bugs?" and features a yellow background with a blue header and footer. The title is in bold black text. Below the title is a bulleted list of four items: "Review", "Testing" (highlighted in blue), "Formal specification and verification", and "Use of development process". The footer contains the IIT Kharagpur logo and the text "NPTEL ONLINE CERTIFICATION COURSES 199".

- Review
- **Testing**
- Formal specification and verification
- Use of development process

Assuming that the best programmers make mistakes and there lead to bugs the code, how do we reduce bugs? Because finally, we have to give a good software to the customers, there are many techniques to reduce bugs one is to do review. Can do a specification review, design review, code review, and that is a very effective way to reduce the bugs.

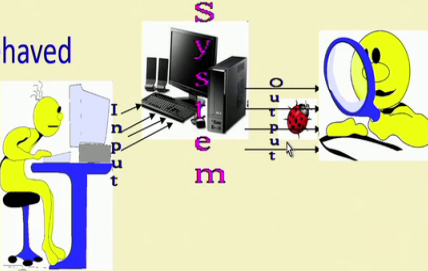
Testing, that say widely practices and acknowledged to be a good technique to reduce bugs. Formal specification and verification this not used for all the part of the code these are expensive difficult to use cannot handle large programs and so on. And therefore, their use is be testified use of a proper development process, this a defensive mechanism, it reduces the number of bugs. In this lecture today's lecture we will look at the testing concentrate on testing.

(Refer Slide Time: 07:40)

Page 35/35

How to Test?

- Input test data to the program.
- Observe the output:
 - Check if the program behaved as expected.



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 200

The first question that, need to ask is that how do you test a program? Somebody may even ask you this question that how do you test the program? We test a program by giving some inputs to it called as the test inputs. And, then observe the output and see if the output matches our expectation, bit matches we said at that test cases passed, but has failed we note down for which data it failed and under what conditions?

So, input data to the program observe the output and check if the program behaved as expected give inputs to the system, observe the output and see if it is exactly is per our expectation. It is exactly is per our expectation, then that is passed, but there is a discrepancy there is a failure. And, we note down the conditions that is what input we gave under what conditions?

(Refer Slide Time: 08:50)

Examine Test Result...

- If the program does not behave as expected:
 - Note the conditions under which it failed (Test report).
 - Later debug and correct.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 201

If, the program does not behave as expected, we note the conditions under which it failed and this we call as the test report. And, this test report based on the test report debugging done to identify the exact faults or the bugs and then these are corrected.

(Refer Slide Time: 09:17)

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?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 202

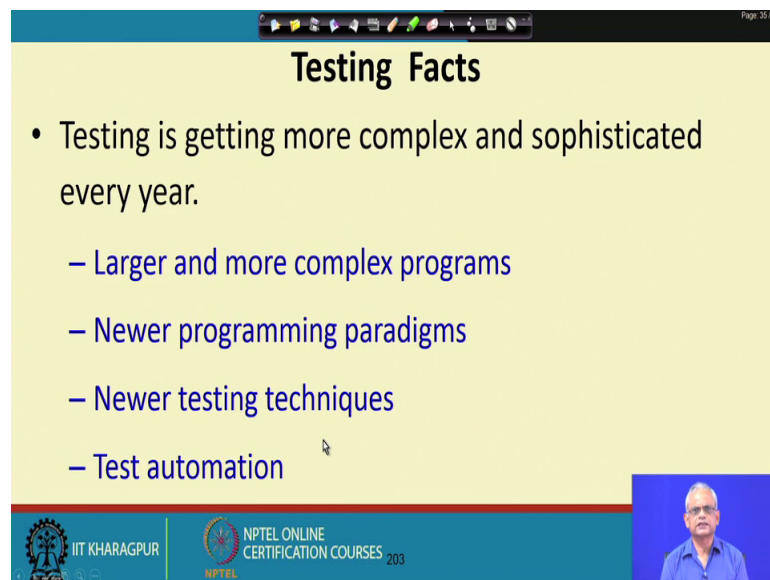
For a typical program in a industry scenario, testing consumes the largest effort among all development phases. And, it also has the largest manpower. So, if you walk into any organisation you will find that there are more number of testers, than there are designers or coders or those who does specification the analysts.

Since, the industry needs a large number of testers implies more job opportunity. Typical estimate is about 50 percent of the development effort is spent on testing. Because finally, we need to give customer a very reliable software, if we deliver a software which is buggy, the company will get a bad name and the company cannot progress.

So, most companies are careful about testing they spend about 50 percent of the development effort in testing, but then testing is done towards the end of the development life cycle. And only 10 percent of the development time is typically taken to carry out the testing, but how does 50 percent of the development effort spent in 10 percent of the development time. The answer is that there is lot of parallelism in testing, many testers can carry out the work the same time, different parts, different test, they can execute different test cases and so on.

And therefore, using a large manpower the testing time is reduced, but there is less parallelism in specification or design, because their work is dependent on each other cannot really parallelly deploy 100 designers or 100 analysts doing the specification.

(Refer Slide Time: 12:01)



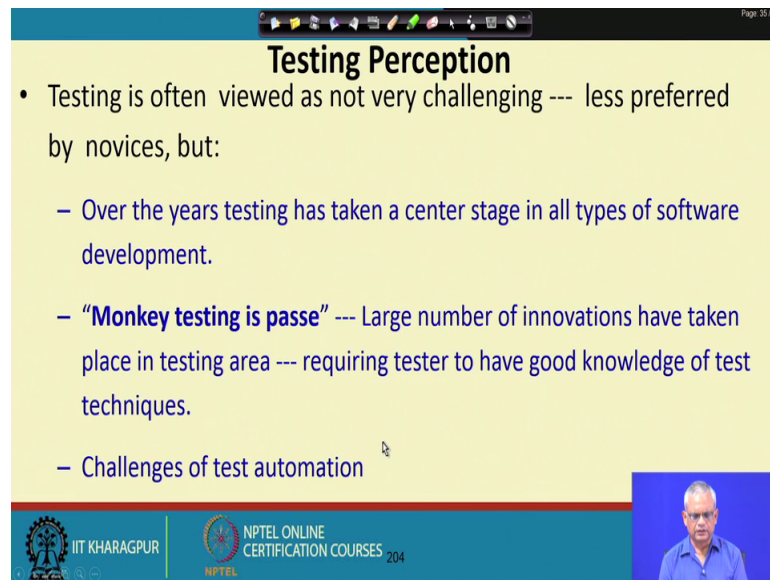
Testing Facts

- Testing is getting more complex and sophisticated every year.
 - Larger and more complex programs
 - Newer programming paradigms
 - Newer testing techniques
 - Test automation

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 2013

Over the years testing is getting complex and sophisticated. The reasons are that the programs themselves are becoming larger and more complex, newer programming paradigms, test automation, there is sophistication you must know how to use the test tools? And, also newer testing techniques many new testing techniques have been developed recently and testers must know these testing techniques.

(Refer Slide Time: 12:43)



The slide is titled "Testing Perception" and contains the following content:

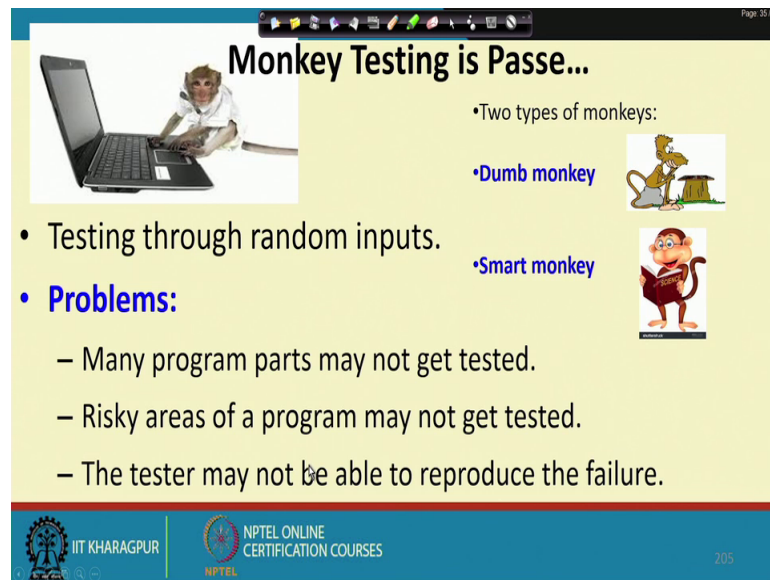
- Testing is often viewed as not very challenging --- less preferred by novices, but:
 - Over the years testing has taken a center stage in all types of software development.
 - **“Monkey testing is passe”** --- Large number of innovations have taken place in testing area --- requiring tester to have good knowledge of test techniques.
 - Challenges of test automation

The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES 2014 logo, and a small video inset of a speaker.

But, those who do not know this they think that testing is not challenging, but over the years testing has taken a centre stage, and it has become much more challenging than even coding or designing or specification. The reason why these perceptions exist is that in the early years of testing, testing was done by inputting random test values, which is called as monkey testing.

Because of the large number of testing techniques and test related innovations tools etcetera that have come in to picture, monkey testing is no more used. And, the testers have their own domain knowledge and not everybody can do the testing. And also they must be convergent with the test tools.



(Refer Slide Time: 13:51)



Monkey Testing is Passe...

- Testing through random inputs.
- **Problems:**
 - Many program parts may not get tested.
 - Risky areas of a program may not get tested.
 - The tester may not be able to reproduce the failure.

•Two types of monkeys:

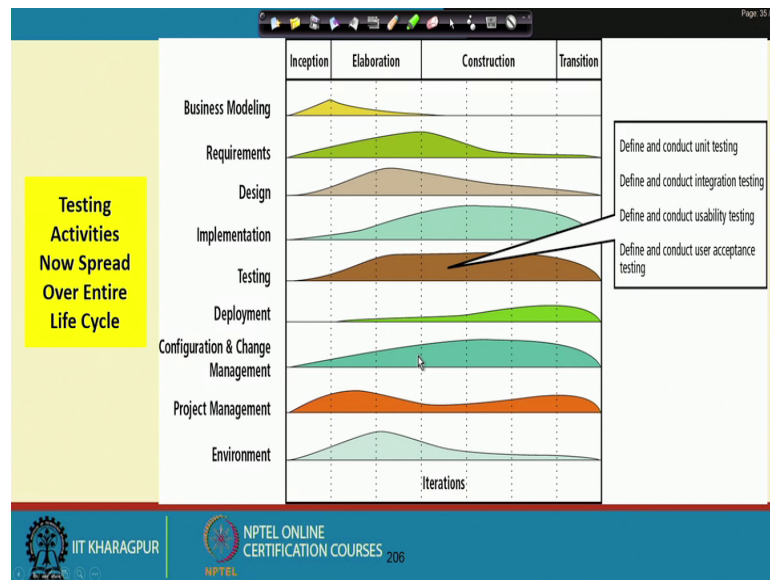
- Dumb monkey 
- Smart monkey 

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 205

In the initially you are sub testing used to be called as monkey testing. Basically give input data and observe it anything happens. And, there were 2 types of monkeys; initially the dumb monkeys. The dumb monkeys they understood very little and just kept on typing giving data. Whereas, the smart monkeys they knew that how the software works? What are the menu choices? What it is menu choice needs to do? They can execute specific scenarios and so on. So, the dumb monkeys could only crash the system, that is the only thing that they can notice whereas, the smart monkey knows what data is expected and so on.

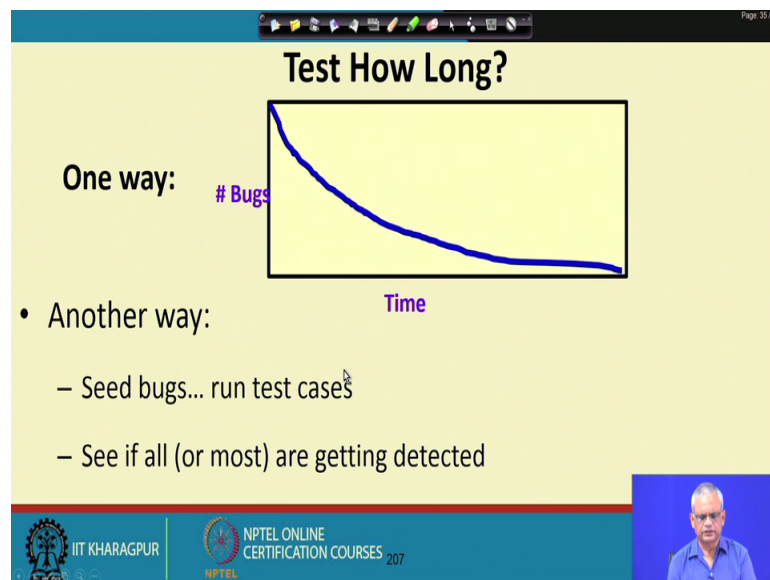
And therefore, is a much more effective tester, but then both the monkeys they give random inputs. The problem with monkey testing is that if random data many parts of the program do not get tested, the risky areas of the programmer not tested well, because they just given random input and not identified the risky areas and tried to test those. And, also the worst thing is that many times they just say that the program failed, but they cannot identify why, what have they been doing? They cannot reproduce the failure.

(Refer Slide Time: 15:43)



This is another basic concept that nowadays the testing effort is spread over the entire lifetime.

(Refer Slide Time: 15:48)

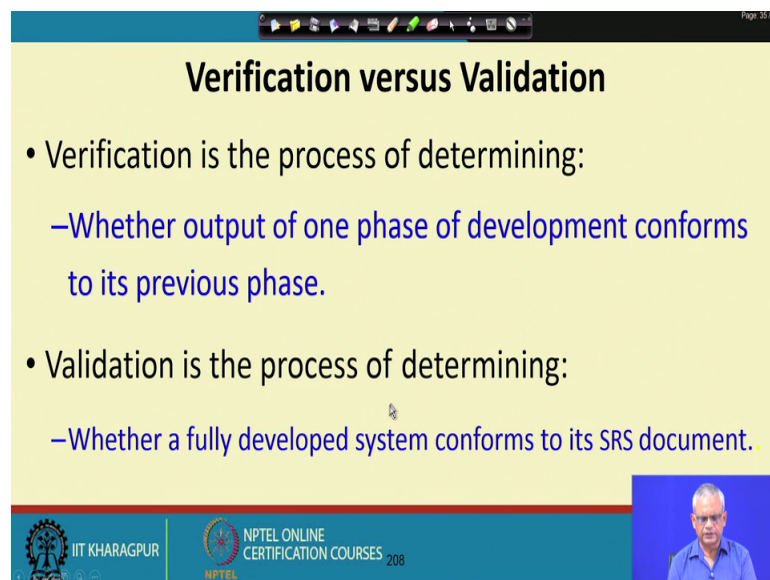


In the water fall model the testing was done towards the end of the life cycle, but now it is spread over the entire lifecycle. Define and conduct unit testing, define and conduct integrate testing, use ability testing, user acceptance testing and so on. But, another basic issue that we must understand before we look at the test methodology then so on is test

how long? Started to test as we test more and more find some failures and So on, but then when do we know when to stop testing that is the stopping criterion.

One way is that as the failure reduces, we test let say we say that we test for 2 days and if we do not find a failure we will stop there. So, that is one way. The other way is that, the manager can seed some bugs and then as the testing proceeds, you will identify whether all those seeded bugs have been found out. For all those bugs have been found out and we will say that possibly the other bugs could have been also been found out and that the time to stop testing.

(Refer Slide Time: 17:26)



The slide is titled "Verification versus Validation" and contains the following content:

- Verification is the process of determining:
 - Whether output of one phase of development conforms to its previous phase.
- Validation is the process of determining:
 - Whether a fully developed system conforms to its SRS document.

The slide also features a navigation bar at the top with icons and a page number "Page 37/38". At the bottom, there are logos for "IIT KHARAGPUR" and "NPTEL ONLINE CERTIFICATION COURSES 208", along with a small video inset of a man in a blue shirt.

This is the, another very basic concept is verification versus validation. Verification is the process of determining, whether the output of one phase conforms to the previous phase. Whereas, validation is the process of determining, whether a fully developed system conforms to its SRS document.

(Refer Slide Time: 17:51)

Verification versus Validation

- Verification is concerned with phase containment of errors:
 - Whereas, the aim of validation is that the final product is error free.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 209

The verification is done after every phase just to check, whether it conforms to the previous phase. We can think of verification as the technique for phase containment of errors. Whereas, validation is for the fully developed software, we check if the final product is error free.

(Refer Slide Time: 18:17)

Verification and Validation Techniques

- Review
- Simulation
- Unit testing
- Integration testing

- System testing

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 210

What are some of the verification techniques? These are review, simulation, unit testing, and integration testing. Just observe here, that unit testing and integration testing are verification techniques, because these are not done on the fully developed system. The

system testing is done fully on the fully developed system. And therefore, the system testing is a validation technique whereas; unit and integration testing are verification technique.

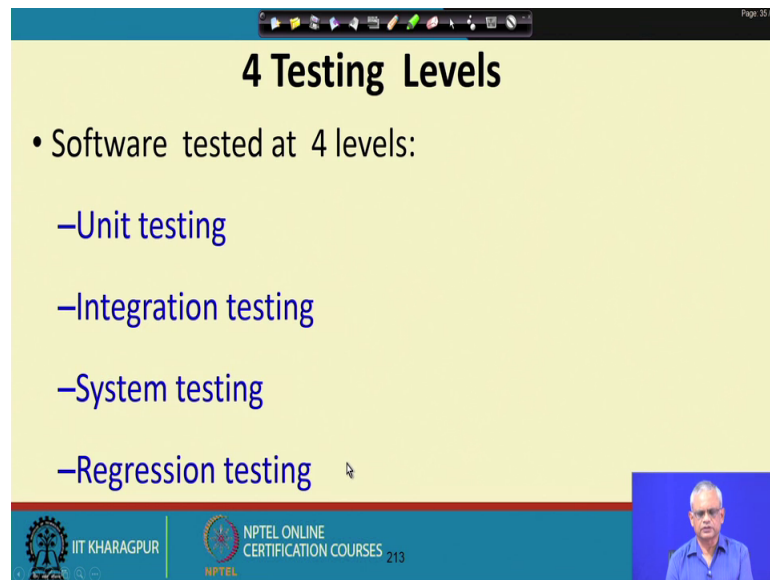
(Refer Slide Time: 18:54)

Verification	Validation
Are you building it right?	Have you built the right thing?
Checks whether an artifact conforms to its previous artifact.	Checks the final product against the specification.
Done by developers.	Done by Testers.
Static and dynamic activities: reviews, unit testing.	Dynamic activities: Execute software and check against requirements.

We can make further distinction between verification and validation, verification we check whether we are building the system right that is not committing any errors try to detect the errors as quickly as possible. Whereas, validation we check that after the final thing we have built, whether we have build the correct thing. Here in verification we check the artefacts that are developed after a faith if it conforms to the previous artefact.

Whereas, validation checks the final product against the specification the verification activities are done by the developers. Whereas, the validation by the testers. Verification can be static or dynamic activity in static activity you do not need to execute the program. For example, review is a static activity or we might have to actually run the program and that is a dynamic activity unit testing is a dynamic activity. Whereas, validation is always a dynamic activity, we execute the software and check against the requirements.

(Refer Slide Time: 20:21)



Slide 213: 4 Testing Levels

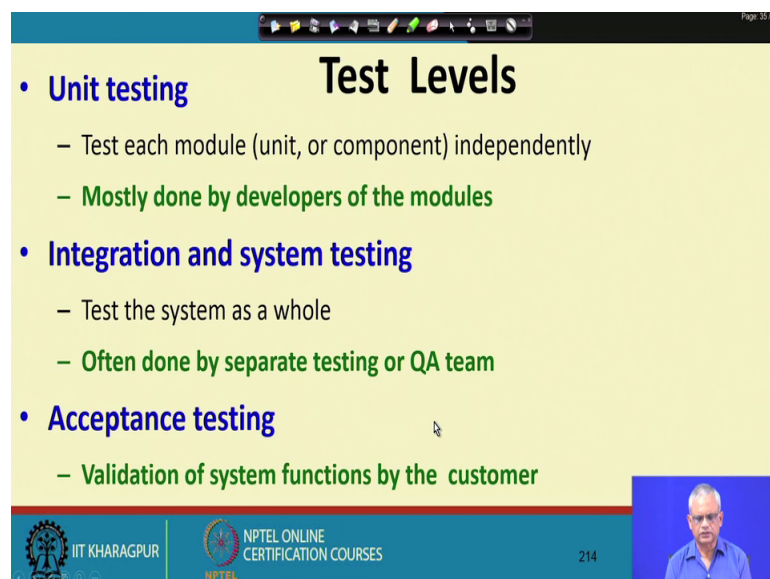
- Software tested at 4 levels:
 - Unit testing
 - Integration testing
 - System testing
 - Regression testing

Page 35/35

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 213

Now, let us look at the testing levels. Software is tested at 4 levels; unit testing, integration testing, system testing and regression testing. Unit testing each unit is tested, that is each module or function, integration testing a set of modules is integrated and tested the system testing all the modules after integration that is complete system is ready tested. In regression testing is done during maintenance any bugs that are fixed need to do the regression testing.

(Refer Slide Time: 21:05)



Slide 214: Test Levels

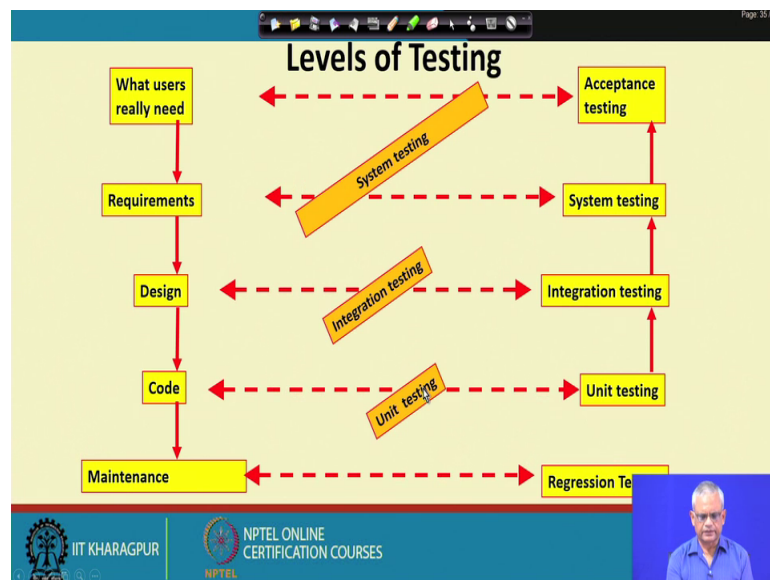
- **Unit testing**
 - Test each module (unit, or component) independently
 - **Mostly done by developers of the modules**
- **Integration and system testing**
 - Test the system as a whole
 - **Often done by separate testing or QA team**
- **Acceptance testing**
 - **Validation of system functions by the customer**

Page 36/36

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 214

Unit testing each module unit that is a function or component is independently tested, mostly done by the developers of the module. Whereas, integration and system testing are done by the testing or a quality assurance team and acceptance testing is a validation testing done by the customer.

(Refer Slide Time: 21:38)



From this diagram, you can see here that on the development side, we find what the user needs, get the requirement, then design and code finally, maintenance and the testing side, unit testing on the code, integration testing at the design and system testing on based on the requirements and regression testing for the maintenance work.

(Refer Slide Time: 22:14)

Overview of Activities During System and Integration Testing

- Test Suite Design
- Run test cases
- Check results to detect failures.
- Prepare failure list
- Debug to locate errors
- Correct errors.

Tester (responsible for: Test Suite Design, Run test cases, Check results to detect failures., Prepare failure list)

Developer (responsible for: Debug to locate errors, Correct errors.)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 216

If, we look at the testing activities that are done during testing; one is test suite design running test cases checking results to detect failures. And to prepare the failure list or the test report is done by the tester whereas, the developer debugs and correct the error.

(Refer Slide Time: 22:43)

Quiz 1

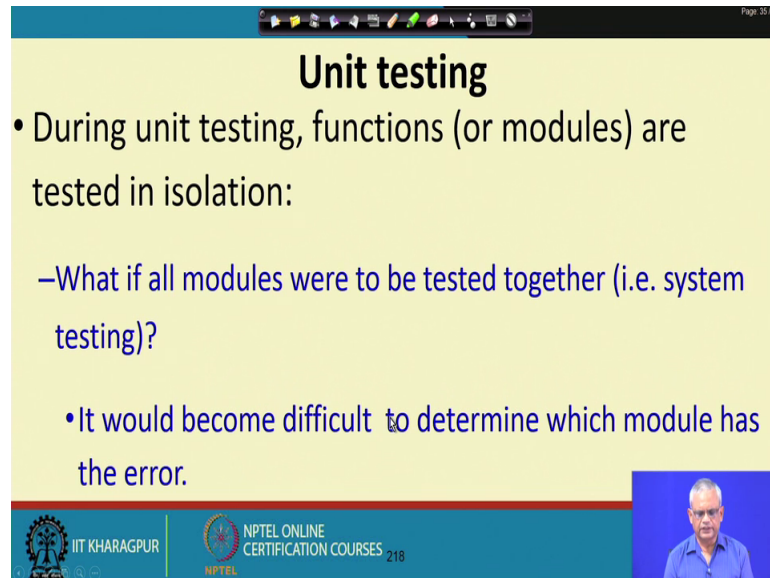
- As testing proceeds more and more bugs are discovered.
 - How to know when to stop testing?
- Give examples of the types of bugs detected during:
 - Unit testing?
 - Integration testing?
 - System testing?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 217

Now, let us have a few quiz as testing proceeds more and more bugs are discovered. So, how do we know when to stop testing? This well said there are 2 techniques; one is to check that few days of testing does not discover any new failures. And the second is by seeding bugs.

Give examples of the types of bugs detected during unit testing, integration testing, and system testing. In unit testing, we can identify the bugs in a unit for example, logical errors. Integration testing, we identify the bugs that are at the interface of 2 modules. So, here we identify the interface bugs, system testing here we identify bugs for example, performance related bugs, which are not identified during unit or integration testing.

(Refer Slide Time: 24:00)



Unit testing

- During unit testing, functions (or modules) are tested in isolation:
 - What if all modules were to be tested together (i.e. system testing)?
 - It would become difficult to determine which module has the error.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 218

Now, let us first look at the unit testing. Here the units are tested in isolation, the units can be functions or modules, but the question is that why not test all the modules together? Why do unit testing? The reason is that if we do not do unit testing, it would be difficult to determine which module has the error in unit testing we are testing only a small unit and we know that the bug is there we quickly debug and correct. But if we do not do unit testing for each bug we have to trace out which unit has the problem and debug it that correct it.

(Refer Slide Time: 24:53)

Integration Testing

- After modules of a system have been coded and unit tested:
 - Modules are integrated in steps according to an integration plan
 - The partially integrated system is tested at each integration step.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 219

So, it becomes much more expensive, if we do not do unit testing. In integration testing we integrate few modules together and then check their interfaces, if there are any problems.

(Refer Slide Time: 25:05)

Integration and System Testing

- **Integration test evaluates a group of functions or classes:**
 - Identifies interface compatibility, unexpected parameter values or state interactions, and run-time exceptions
 - **System test tests working of the entire system**
- **Smoke test:**
 - System test performed daily or several times a week after every build.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 220

In integration testing, we integrate a group of functions or classes and then find if their compatible, if there are unexpected parameter values. On the other hand system testing we look at the entire system, there is another term here smoke test. Before, we start to do a test normally do a smoke test, just imagine that a plumber has put a pipeline. And

before he actually puts real water in that and tests we would like to see there are any leakage. And, they do a smoke test see if there are leakage they put smoke, similar thing here in software in smoke test we just check if some basic functionalities are working and it does become test worthy. We have looked at some very basic concepts in testing we are at the end of this lecture we will stop here and continue in the next lecture.

Thank you.