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

Lecture - 02
Levels of Testing

Welcome to this session. In the last session, we had discussed about some very introductory topics in Testing. Now will continue from that point we will look at few more basic concepts in Testing.

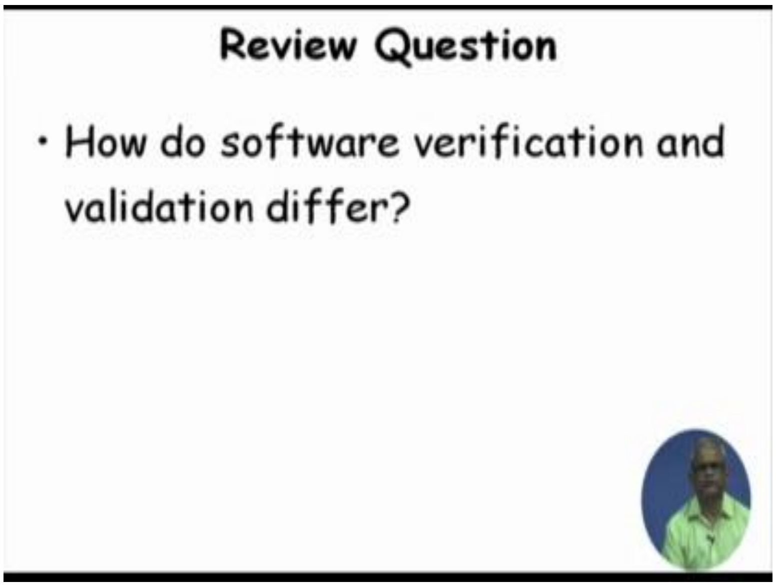
(Refer Slide Time: 00:35)

Testing Levels and Models

6/3/2016 1

Namely - the different Testing Levels and the Life Cycle Models per testing.

(Refer Slide Time: 00:48)

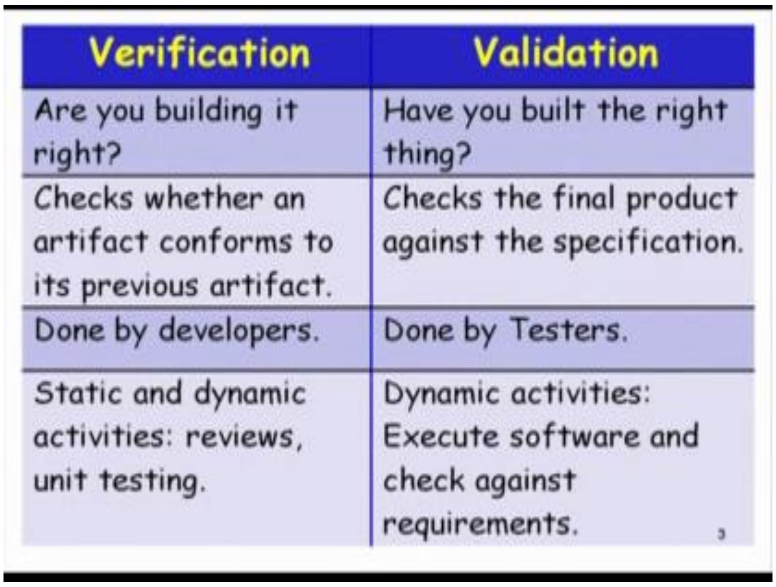


Review Question

- How do software verification and validation differ?

Before we get started will just ask one question based on what we were discussing in the last session. In what ways Verification and Validation of software differ? What are the main differences between Verification and Validation?

(Refer Slide Time: 01:11)



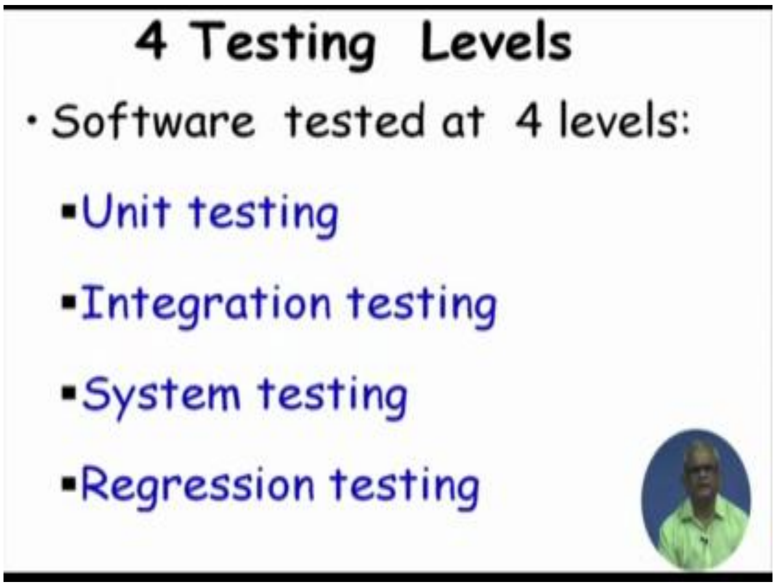
Verification	Validation
Are you building it right?	Have you built the right thing?
Checks whether an artifact conforms to its previous artifact.	Checks the final product against the specification.
Done by developers.	Done by Testers.
Static and dynamic activities: reviews, unit testing.	Dynamic activities: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

3

To answer this question, we can first define that verification concerns about whether we are building the software right? That is checking whether it conforms to its previous artifact. On the other hand validation is concerned about checking whether we have built the right thing, the right software. And here, you check the finally produced software against the specification of the software. And verification, these activities are done during the different development stages and are done by the developers.

Whereas validation is done at the end in system testing and this is done by the testers. Verification techniques are review, analysis and stimulation and so on. So, these are static and dynamic activities. Review analysis etcetera are static activities and it is also a dynamic activity because of the unit testing. Whereas, validation is only dynamic activity it is not a static activity, we do not really need to look at the documents analyze them and so on, need to just execute the software and check against the requirements.

(Refer Slide Time: 02:46)



4 Testing Levels

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

Now, what are the different levels at which the testing is done? Actually, there are 4 levels at which testing needs to be done. One, is at that the Unit level. As the software getting developed unit needs to be tested as soon as it has been developed. Then after all the units have been tested need to do the Integration testing, where we integrate the different units and then test if the integration they are working all right.

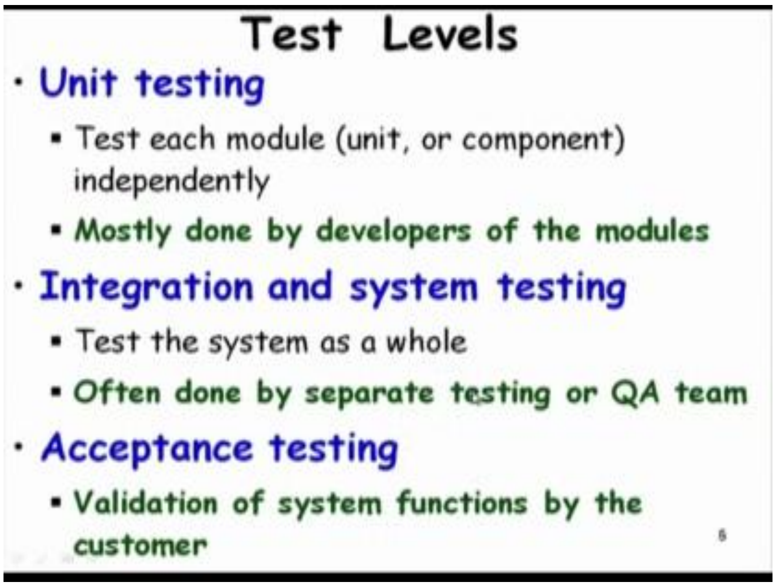
If they are working at the unit level, why should not they work at the integration level? The answer is that, even though the unit may be working all right, the different units, but when you integrate them they may not be interfacing all right. For example, ( ) the specific set of values etcetera may not be getting interchanged properly. The integration testing is done after the unit testing and we check whether the units are integrating or they are working together well.

Then finally, the System testing - where all the units have been integrated and tested against the specification to check if the software is working as you are specified. But then there is another level of testing which is Regression testing. This is undertaken during maintenance, so after the software is released there can be bug reports or enhancement reports, putting requirement and so on and the software gets changed.

Software after it has been completed development it undergoes changes, every software undergoes change. And whenever there is change, we need to check whether the changed part is working all right, not only that we need to check whether the other parts which have not being changed whether they are continuing to work all right.

Why will the parts that were unchanged stop working after a change, because a change might have influence on other parts due to say at data. And this is what the regression testing tries to do is check whether software which has previously passed some test cases, whether after a change it is continuing to pass those test cases. These are called as Regression bugs, where it worked correctly when it was released, but then on a small change somewhere many of the tests ( ) failed and these are the regression errors which cause them to fail.

(Refer Slide Time: 06:21)



Test Levels

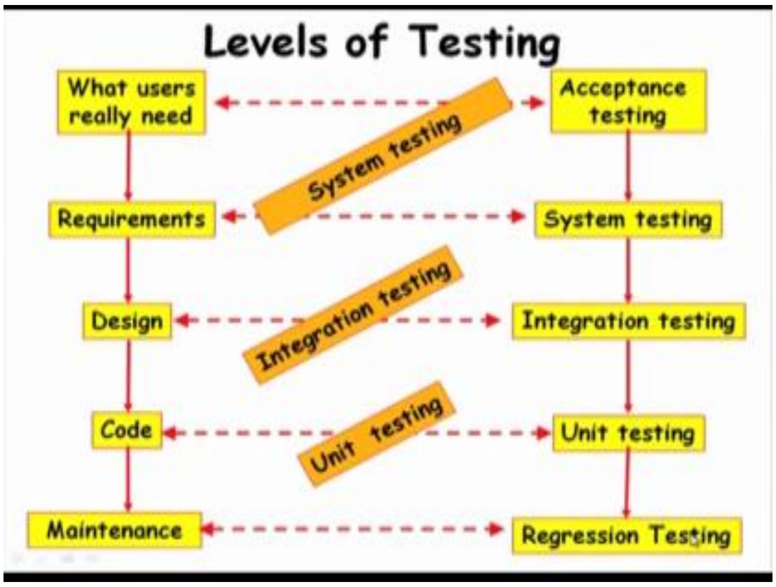
- Unit testing
  - Test each module (unit, or component) independently
  - Mostly done by developers of the modules
- Integration and system testing
  - Test the system as a whole
  - Often done by separate testing or QA team
- Acceptance testing
  - Validation of system functions by the customer

6

So, if we look at the test levels in unit testing, we test each unit. The unit can be a component, it can be a function, or it can be a module. Each of these we can call as a unit, we can call a function as a unit, we can a module as a unit or a component as a unit and we test them independently of other units. And as we were discussing unit testing is done by the developers themselves who were developing the unit.

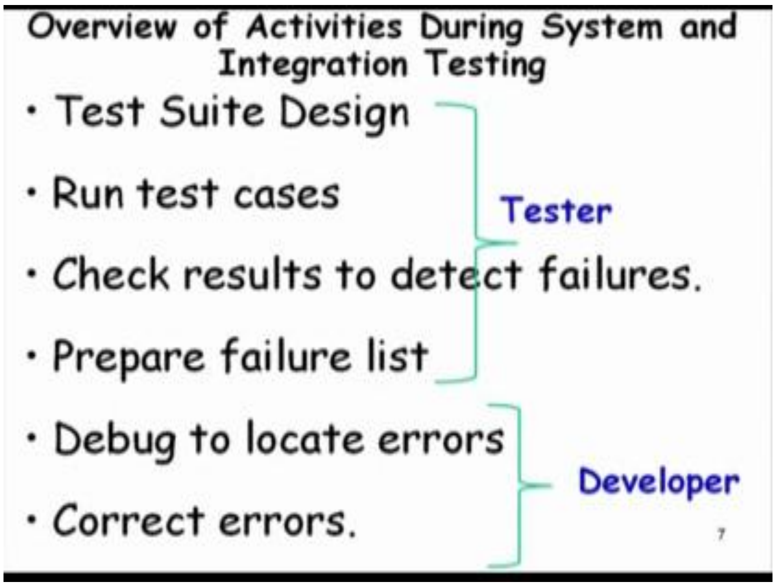
Whereas, in the system testing it is a separate test team w hich typically does the system testing. Here, the test system is tested as a whole. Acceptance testing is a part of system testing and it is validation of the functions by the customer. As you can see here that the system testing or the validation testing is done both by the testers and also by the customers.

(Refer Slide Time: 07:32)



Here, we show diagrammatically that what are the levels of testing. So, the customer, the things that he needs, what the users really need and they conduct the acceptance testing based on that. Whereas, the system testing is based on the requirements both are system testing. Whereas, the integration testing is based on the design, so different units are identified in the design and how do they interface. And based on the design the integration testing is carried out. Whereas, based on the code of a unit, the unit testing is carried out. And during maintenance the regression testing is carried out. These are the different levels of testing.

(Refer Slide Time: 08:27)



Overview of Activities During System and Integration Testing

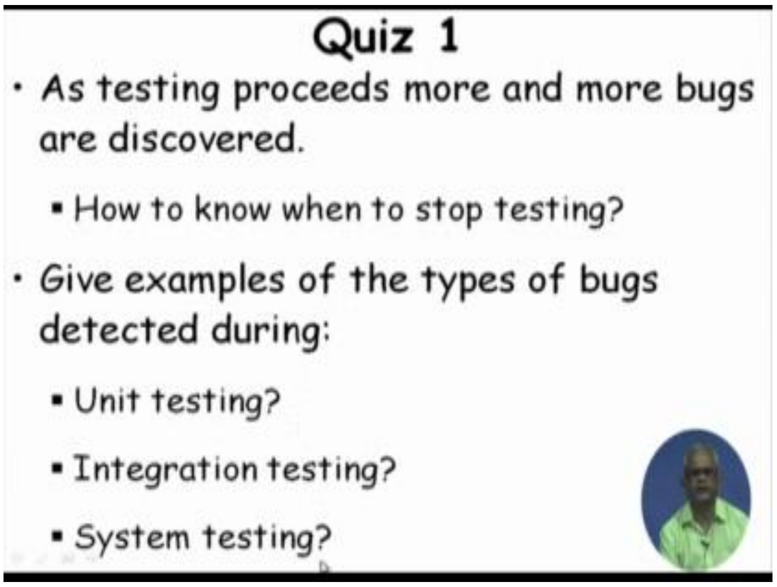
- Test Suite Design
- Run test cases
- Check results to detect failures.
- Prepare failure list
- Debug to locate errors
- Correct errors.

Tester
Developer
7

But then, what are the activities for testing? What activities are undertaken by a tester? And who carries out those activities actually? So let us see the activities that are typically taken the test place for testing software; one is Test Suite Design, Run test cases, Check results of the running of the test case and see if there are failures, and then prepare the failure list or the test report.

So, during the system and integration testing, the testers carry out these. And of course, during unit testing testers are not there all these activities are carried out by the developer himself. Once the failure read list or the test report is prepared it is given to the developers who use the test report to first debug to find out what where the specific bugs in the code or the faults and then the correct those.

(Refer Slide Time: 09:47)



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?

Now, let us try to answer few questions. Already we have discussed this in the first session just for reviewing that whether things have been understood. As testing proceeds more and more bugs are discovered, so when does one decide that there is no further necessity of testing. This we had discussed in the last session and we said that really depends on the type of the software and the specific application that is in mind, but then either based on the rate of bug detection initially large number of bugs are detected per unit time and the rate of bug report falls with time.

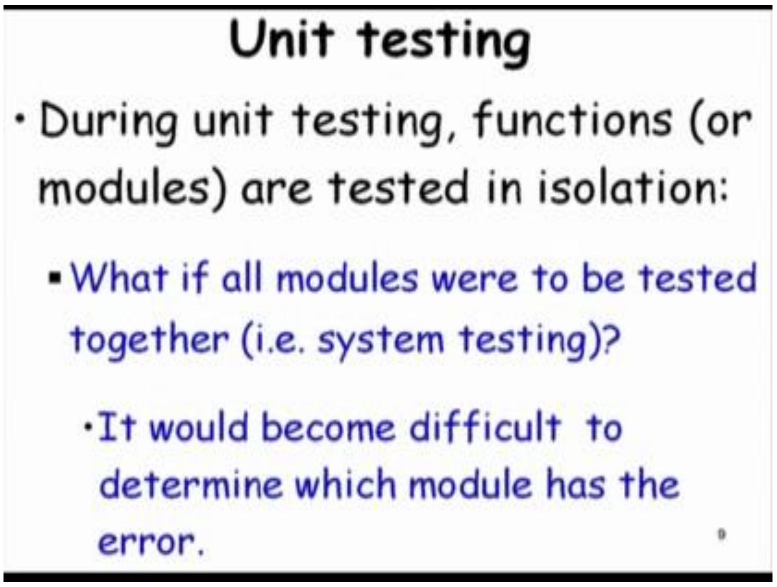
And, once in a day or two of testing no further bugs are reported that is the time to test. And the other way is to seed bugs and check whether those bugs have got detected, and the percentage of bugs that have got detected indicates the thoroughness of the testing, the thoroughness with which the work has been tested.

But then, one question is that the bugs that are seeded should match with real bugs, because if we seed only very trivial bugs which are easily detected then that may not correspond to the how many finally the bugs are remaining. So, the percentage of bugs for each category of defect should match with what a real program has. So, this point we will elaboate further as we proceed, that for bugs seeding we have to be careful you cannot just seed any arbitrary bugs. The type of bugs you seed and the percentage of those should match with an actual program scenario. Now, another question is t hat give some example of bugs that are detected during unit testing, integration testing and system testing.

So, what is a bug that will be detected during unit testing? The answer is, can detect an algorithm error or a programming error, a programmer possibly inter change to variables instead of i he wrote j or maybe he did not implement the algorithm correctly. So, these are types of bugs that are detected in unit testing; but what about the integration testing? One example of a bug detected during unit testing is miss match of parameters. In place of i and j are two different variables, int variables instead of passing i and j it is passing j and i. So it has just interchanged the two variables that were expected. That is a kind of integration bug.

But what about system bug? What kind of bugs will be detected by system testing? One can be a performance bug; of course the other bugs can also exist which have escaped unit testing or integration testing. But my question is that unit and integration testing have been perfectly carried out, and then what bug will be detected during system testing? The answer is performance bugs, because unit and integration do not look at performance aspects, usability aspects, so these are detected during system testing.

(Refer Slide Time: 14:07)



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.

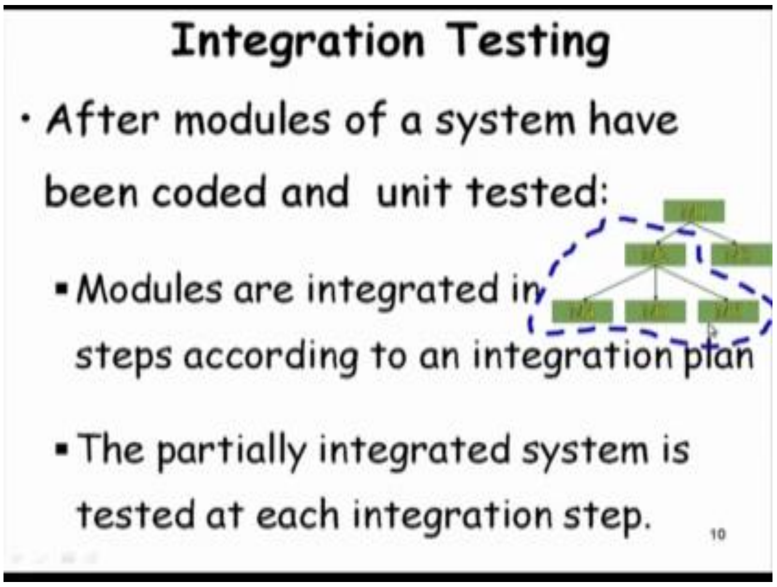
Now, let us look at some very basic uses in unit testing. We are saying that this is the first level of testing where a unit is tested, and a unit can be a function a module or a component. And this is tested in isolation that is different units are tested separately.

But then the question that naturally arises is that, cant we just test all the units together? Why test them separately? Or in other words, why not perform a thorough system testing? Why do we have to test unit wise and then do a system testing, cannot we just do a thorough system testing? The answer is that yes, somebody can do that but it will be very expensive. The cost of testing will increase enormously.

Why is that? Now let us see the answer. The answer is that the cost will increase ( ) enormously because when you test the system which may having thousands of modules and you detect the failure to correct that failure you would have to first locate where the bug is, you have to debug and then you have to look through all that thousand modules trying to find out where the bug is.

So debugging becomes very difficult, debugging and correcting becomes very difficult. No one directly non trivial software bugs are detected during system testing. They are largely detected during unit testing, because unit testing is a very cost effective way of reducing bugs, system testing is very expensive proposition to reduce bugs.

(Refer Slide Time: 16:27)



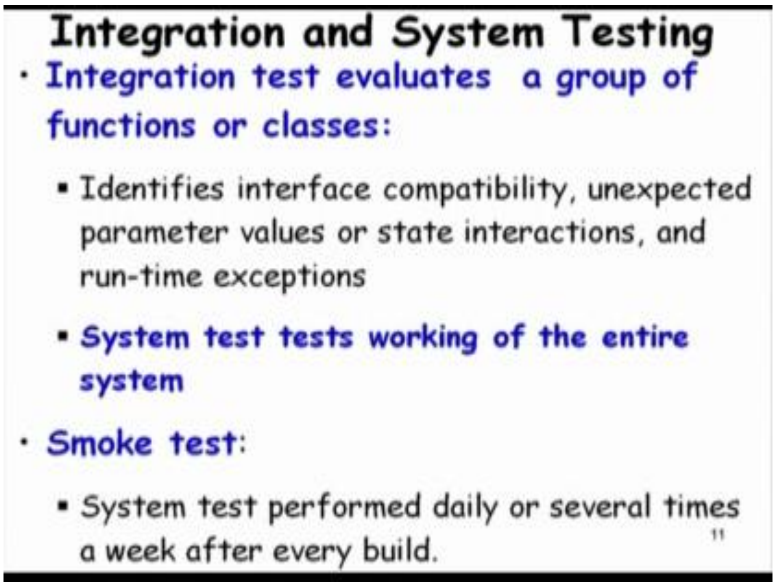
Integration Testing

- 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.

10

Now unit testing, of course will see later more detail, but then the different modules which are unit tested they are integrated together and then again tested if whether after the integration unit is continuing to work fine. And as you are saying they may not work because of the interfacing errors and we will later discuss about the integration plans and so on, how does one decide which modules to integrate together.

(Refer Slide Time: 17:15)



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.

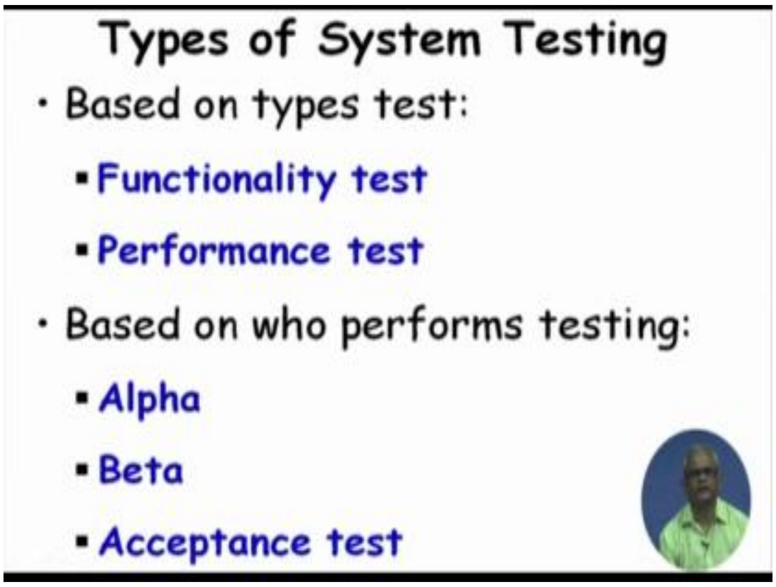
11

As you are saying in integration testing we just integrate groups of modules,functions or classes, whereas system testing we check the entire system. But then every organization recommends that smoke testing be carried out frequently, performed daily or several times a day. Even though system testing is not done or integration testing is not done these are done towards the end of an iteration, but smoke testing is done frequently. Why is that? A smoke testing is actually performing a build of the system that is putting all the modules together and checking whether at least the basic functionalities are working. Otherwise, it will indicate a very severe integration problem.

And if you do not do a smoke testing later integration will be a big problem, integration and system testing because you find that you have integrated everything in system testing the system is just frozen it is not responding to anything, so we do not want that. Need to do frequent smoke test.

The smoke test gets its name from testing pipes and so on, where they use smoke to check before they finally put water and test they would put smoke and large pipe installations and check whether at least the smokes are not escaping before they can do the actual testing. Here, in software context unless we do frequent builds and do a smoke test, at the end we might have big problems and testing will take ( )unduly long time and effort.

(Refer Slide Time: 19:27)



Types of System Testing

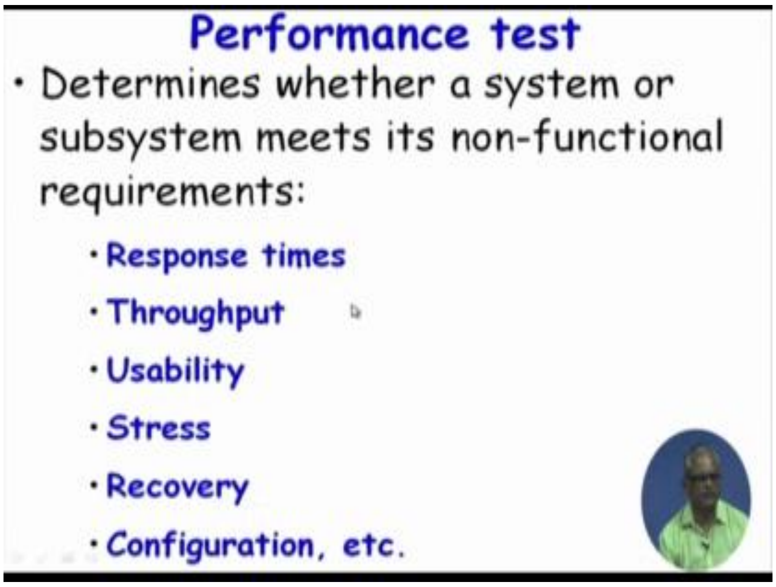
- Based on types test:
  - Functionality test
  - Performance test
- Based on who performs testing:
  - Alpha
  - Beta
  - Acceptance test

We will discuss more about system testing later. But as a brief overview in the system testing we have the entire set of modules integrated and we have the system that is working. How do we do the system testing? We test two types of things; one is whether it is the functionalities are correct and also the performance is correct. So, there are two major types on system testing, the functionality test and performance test.

And also we'll see that there are actually three types of system testing depending on who does the system testing. If it is develop the development team or the developing organization who does the testing is called as the Alpha testing if it is a friendly set of customer it Beta testing and if it is the customer who finally before accepting delivery of the product does the test that is called as the Acceptance testing.

So system testing, three types and depends on who does the testing. Alpha testing is done by the developing organization and here as we can see that they have access to the internals of the software. Whereas, beta testing done by friendly customers, we just offered them please use the software and check if it is working all right and there are bugs. And after this, finally it is the customer who takes, before taking delivery of the software does the acceptance test.

(Refer Slide Time: 21:16)



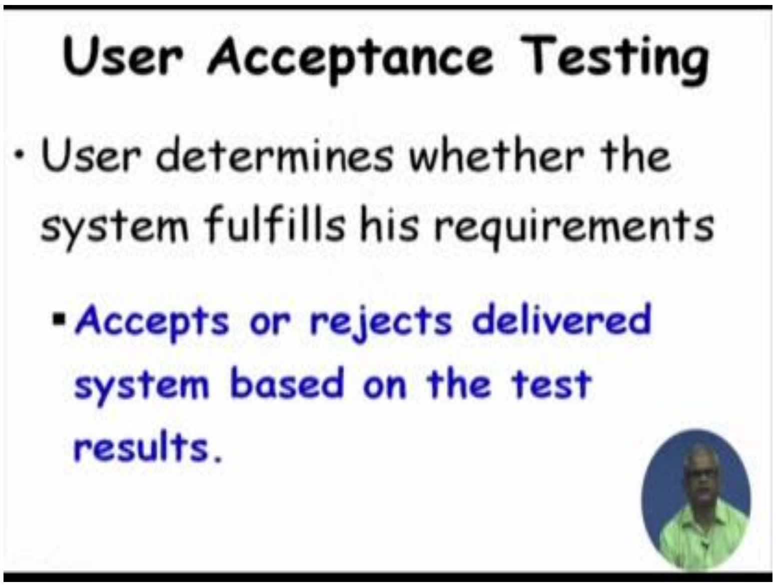
Performance test

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

The Performance tests are actually many types as we will see later in more detail; the performance test deal with checking the non functional requirement. The functional tests, they check the functional requirements in the SRS document, the requirements document.

Whereas the performance tests, they check the non-functional requirements as documented in the requirements document. There are many types of performance tests for example; Response times, Throughput, Usability, Stress working under stressed conditions like number of inputs per unit time is large and so on, Recovery, Configuration. So we will see these aspects in more detail later.

(Refer Slide Time: 22:16)

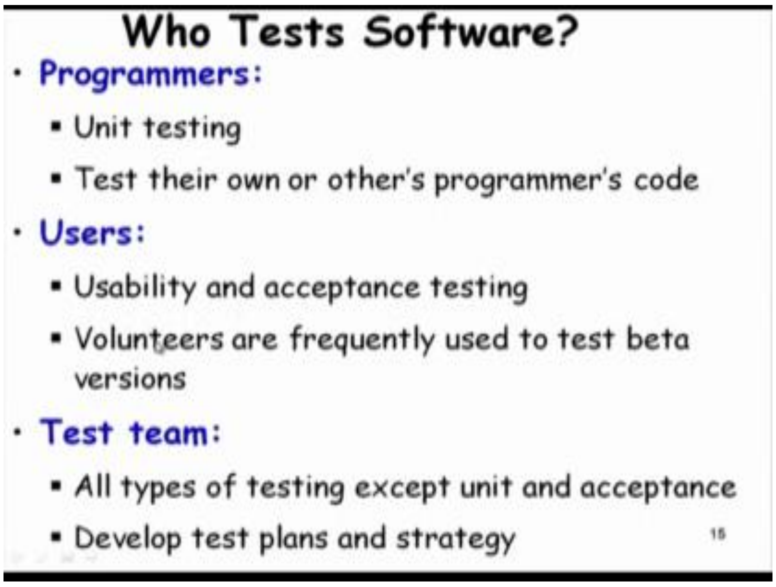


User Acceptance Testing

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

And finally the User Acceptance Testing, this is also a type of system testing. Based on this the user either accepts or rejects the delivered software.

(Refer Slide Time: 22:20)



Who Tests Software?

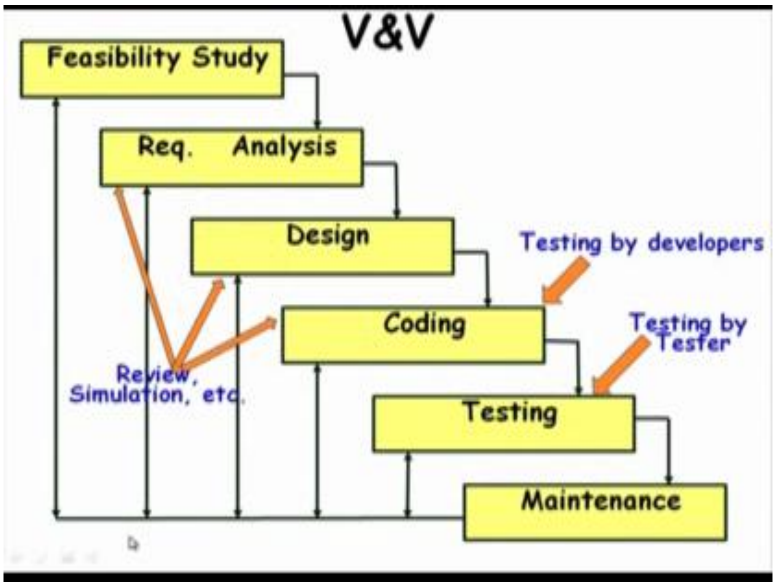
- Programmers:
  - Unit testing
  - Test their own or other's programmer's code
- Users:
  - Usability and acceptance testing
  - Volunteers are frequently used to test beta versions
- Test team:
  - All types of testing except unit and acceptance
  - Develop test plans and strategy

15

Now, let us try to answer this basic question. What are the different types of persons who do testing? One is the Programmers, they do testing. What sort of testing they do? They do unit testing. The code after they develop, they do testing at the unit level. The Testing they do all types of testing that is integration and system testing excepting the unit and acceptance testing they are involved in all other testing.

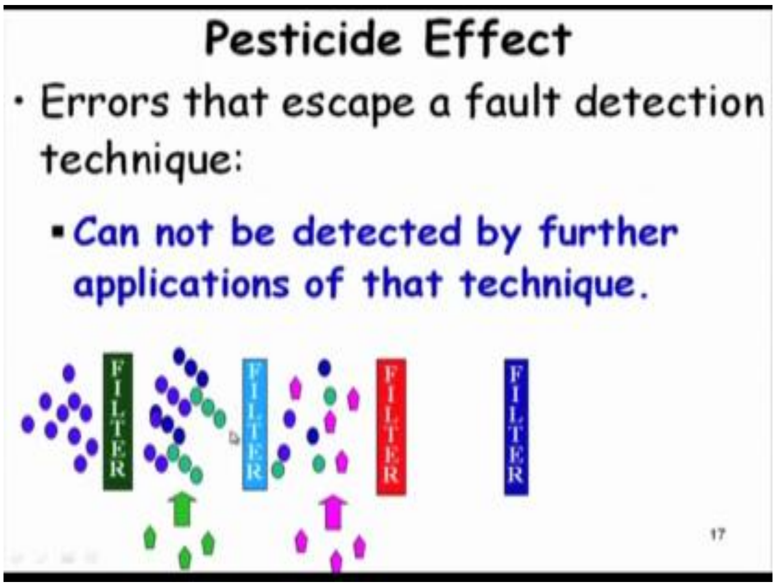
And also they develop the test plan and strategy. The users they look at the acceptance testing, they perform the acceptance testing and also usability. And some of them they may volunteer for the beta testing.

(Refer Slide Time: 23:16)



Now, if we represent these activities on a water fall model we can see that, the test team is largely working during the testing phase. They are not needed elsewhere just see here testing the testers are not needed large number of testers needed here. So, that is a problem with the water fall model. What do the testers do other times? Possibly work in other projects and so on. During coding unit testing is done by the Developers. And the Review Simulation etcetera, the verification activities they take place in the development phases and these are also carried out by the developers.

(Refer Slide Time: 24:23)



Pesticide Effect

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

Now, before we start discussing more details about testing, we need to discuss about another very basic issue about testing that goes by the name Pesticide Effect, because of its analogue to use a pesticide in a crop. Just assume that you are a farmer and you planted some crop, let us say cotton crop and then the cotton crop has grown up and then you find that it is infested with the bugs, pests.

So what do you do possibly by DDT spray? The DDT would kill the bugs, most of them will get eliminated, but may be some 20-30 percent may survive, and then you plant the cotton crop next time and then you find that again bugs are there, so you try spraying DDT it does not work. The bugs have got immune to the DDT. You buy a new pesticide may be Malathion or something, and then spray again 30 percent of the bugs still survive to the next generation.

We have analogy with that in software testing so that is called as a Pesticide Effect. The analogy says that the bugs that escape detection by some fall detection technique cannot be detected by further application of that technique. Just like the bugs that had got immune to DDT could not be killed by DDT anymore, the same thing happens here. In testing we have many testing techniques we can call them as Filters. Initially there are some bugs and we apply one type of filter, may be let us say equivalence partitioning base test and then we find that some bugs are those bugs largely been eliminated only 20 30 percent surviving.

But then as those bugs are fixed and new development takes place we have different types of bugs that have come in. And if we applied the same technique here the bugs that have escaped will not really get eliminated. So, the equivalence partitioning test, if we keep on applying the same test the bugs that have survived will never get eliminated, we have to apply ( ) new tests, maybe decision table testing, may be white box testing, path testing may be MCDC testing.

So, there are dozens of testing techniques and each of them are bug filters. So, the bugs that are survive a filter cannot be eliminated by further application of that filter and also new bugs get introduced by correcting the bugs that where adjusting and also newer development and so on. So, new types of bugs are there. So, as the system development proceeds, we use these filters, and we need to use many types of filters to effectively reduce the bugs. So many test methodologies have to be used, dozens of them.

Just one methodologies used for longtime may not really help. We will just running out of time for this slot; we will continue with this pesticide effect and will do a small problem. So, what is the probability of a bugs surviving by application of a filter? And what is the probability of a bug surviving when we apply 3 or 4 or may be 6 filters one after other. That will discuss in the next session.

Thank you.