# Software Testing

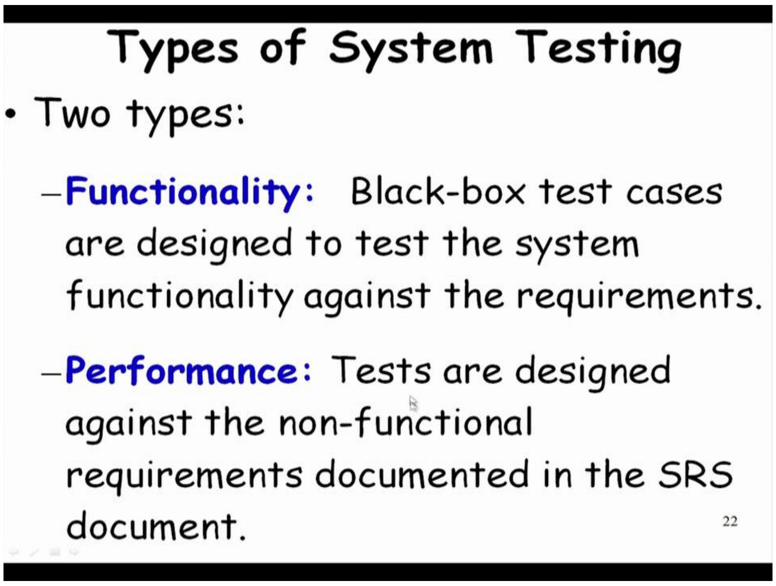
**Prof. Rajib Mall**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**

## Lecture – 17
## System Testing

Welcome to this session. So far, we have looked at basic concepts on testing. And then, later we looked at unit testing, both black box and white box testing. And, the main objective there was to remove the faults as much as possible from a unit. And, we said that the definition of a unit varies; it can be a function or a module. And once a unit, all the units have been fully tested, we do the integration testing.

And, the main motivation behind integration testing is to remove the faults at the interfaces of various units. And, after the integration testing is over, the system testing is taken up, where the full system is tested as a whole. And, here the testing is done against, the SRS document. So far, both unit and integration testing; it is tested as per the design the design document, whether the unit and integration testing alright. And therefore, the unit and integration testing are known as verification techniques. Whereas the system testing, where the final software that is to be delivered to the customer is tested against the SRS document is known as system testing. Now, let us see what is involved in system testing.

(Refer Slide Time: 02:11)



### Types of System Testing

- Two types:
  - **Functionality:** Black-box test cases are designed to test the system functionality against the requirements.
  - **Performance:** Tests are designed against the non-functional requirements documented in the SRS document.

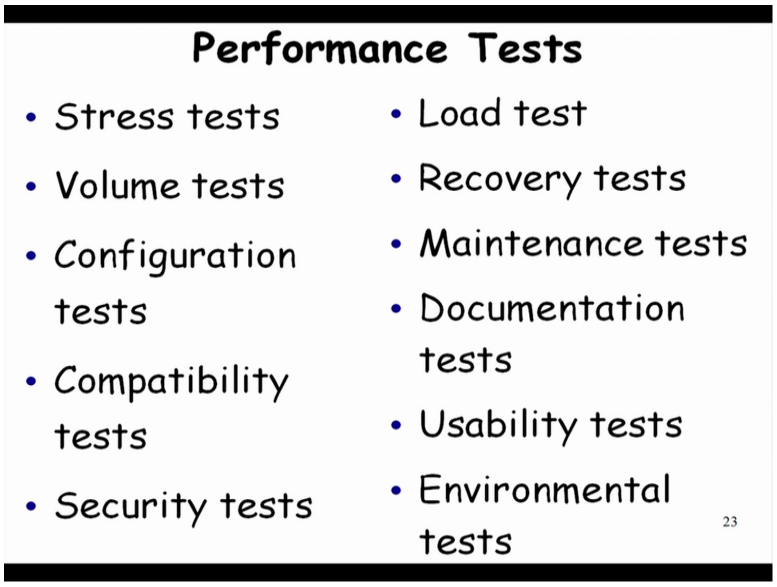
22

Actually we had said so far, in the last session that there are three types of system testing, depending on who carries out the testing. If it is done at the customer site, then it is called as alpha testing. Sorry, if it is done by the developer site, it is known as the alpha testing; if it is done by some friendly customers, it is known as beta testing and if it is done by the customer himself or herself, it is known as the acceptance testing.

So, in all the system testing there are basically two types of tests that are carried out. The functionality; the functionality is tested as specified in the SRS document. And, the functionality is tested as black-box testing because these are the specification of the functions. And, we know how to design black-box tests for a function. We have discussed several techniques for black box testing. And, all these techniques are applicable here for the functionality testing of a system.

The other category of testing which are so far not been tested, neither in unit or integration testing are the performance tests. The performance tests are designed to test the non-functional requirements of the system as documented in the SRS document. So, let us see what are the different categories of performance tests that are done; The black-box functional testing we have discussed so far in quite a bit of depth. And, we do not have to really discuss separately how the black-box test cases are to be designed for system testing because the same concepts are applicable here. Let us look here about the system test, about the performance tests involved in system testing.

(Refer Slide Time: 04:34)



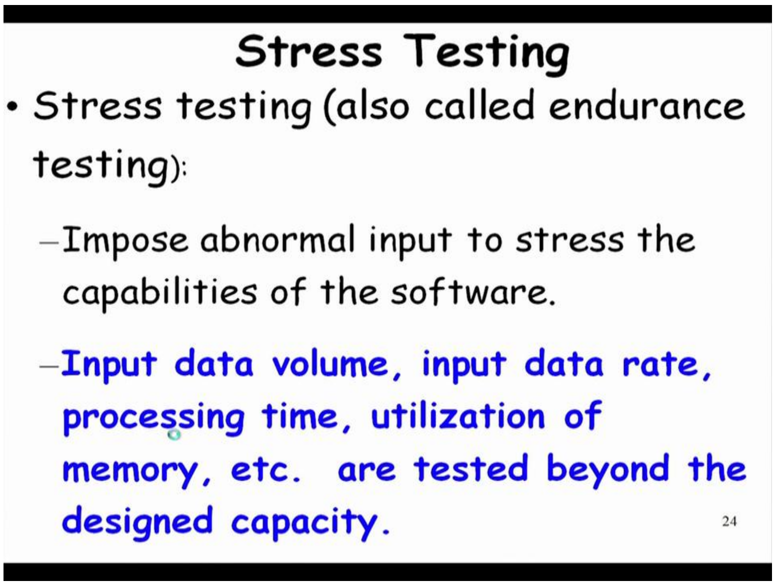
## Performance Tests

- Stress tests
- Volume tests
- Configuration tests
- Compatibility tests
- Security tests
- Load test
- Recovery tests
- Maintenance tests
- Documentation tests
- Usability tests
- Environmental tests

23

There are a variety of performance tests that can be carried out on a system. But, not all of them need to be carried out. It depends on what are the non-functional requirements specified for the software. So, we look at the non-functional requirements. And, based on what is specified for the system, we have to carry out some or all of these various performance tests.

(Refer Slide Time: 05:14)



## Stress Testing

- Stress testing (also called endurance testing):
  - Impose abnormal input to stress the capabilities of the software.
  - **Input data volume, input data rate, processing time, utilization of memory, etc. are tested beyond the designed capacity.**

24

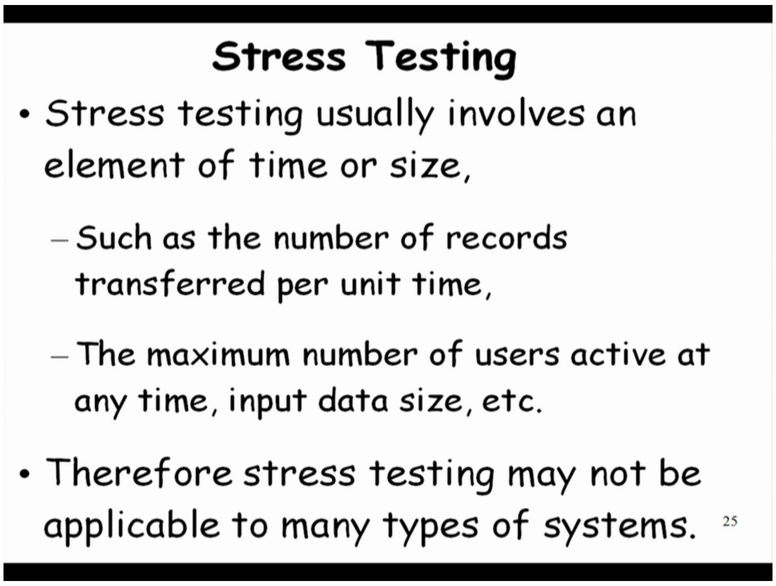
Stress tests; the stress tests is also called as endurance testing. Here, we give abnormal inputs to stress the capability of the software. By abnormal input, we mean beyond what is specified in the SRS document. Let me repeat that because this is a very important statement. In stress testing, we give input to the software that is beyond what is specified for the SRS document. For example, we might give more data volume, the size of the database for a software than what is specified. If it is specified hundred, thousand records we might give hundred, twenty thousand records.

Input data rate; so, if the data rate that the software is required to handle is 100 kilo bytes per second, we might test with 110 kilo bytes per second, just to check is there any abnormal reaction by the system. If it slightly exceeds the specified data rate, does the system crash? Does it behave something very unacceptably because if we gives input which is beyond what is specified, it is expected that the system will show a degraded performance. But, that should be a gracefully degraded performance, rather than a discontinuity of the performance like crash or hang and so on.

The processing time; if the processing time is required to be of few millisecond, we might give slightly more than that, utilization of the memory, etcetera. So, these are tested beyond the designed capability.

So, stress testing is giving input which is beyond what is specified in the SRS document, just to check that the system gracefully degrades. If we give input which is beyond the specified capability. And, there is no discontinuity such as crashes or system hangs and so on.

(Refer Slide Time: 08:22)



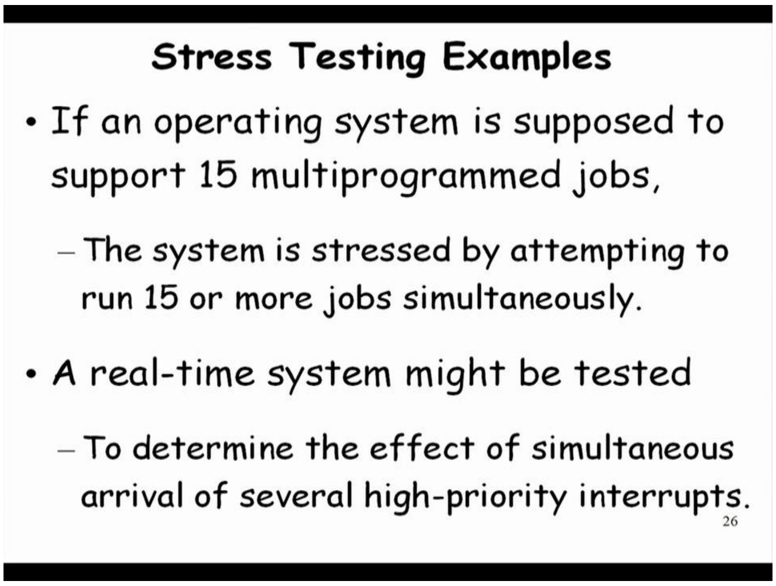
## Stress Testing

- Stress testing usually involves an element of time or size,
  - Such as the number of records transferred per unit time,
  - The maximum number of users active at any time, input data size, etc.
- Therefore stress testing may not be applicable to many types of systems.

25

So, stress testing usually involves an element of time or size. For example, the number of records to be transferred per unit time, maximum number of users active at any time. For example, a system may be designed to support ten active users at any time. Now, during stress testing we have to check that if there are eleven active users. Does the system still function even though in a slightly degraded mode or does it go into a crash or a hang? If it is performing with eleven users, when ten is the specified number of users, in a slightly degraded mode it passes its stress testing. Similarly, we check with the slightly larger data size than what is specified. And, as it is must be clear that for many systems, stress testing may not be applicable.

(Refer Slide Time: 09:32)



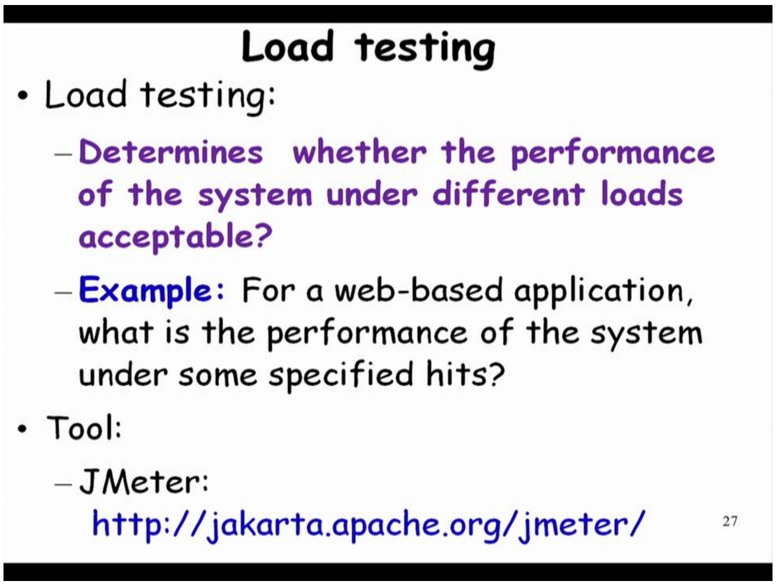
## Stress Testing Examples

- If an operating system is supposed to support 15 multiprogrammed jobs,
  - The system is stressed by attempting to run 15 or more jobs simultaneously.
- A real-time system might be tested
  - To determine the effect of simultaneous arrival of several high-priority interrupts.

26

Let us look at some examples of stress testing. Let us say we are testing an operating system, which can handle maximum of fifteen multiprogrammed jobs. So, as to in the stress testing, we give; if there are more than fifteen jobs that are initiated simultaneously, what is the response of the operating system. Similarly, a real-time system, we might test the arrival of several high-priority interrupts. At the same time, if the system makes an assumption in the requirement document that any time only one high priority interrupt can come or two priority interrupt can come, we do the stress testing by having slightly more number of higher priority interrupt coming at a time.

(Refer Slide Time: 10:35)



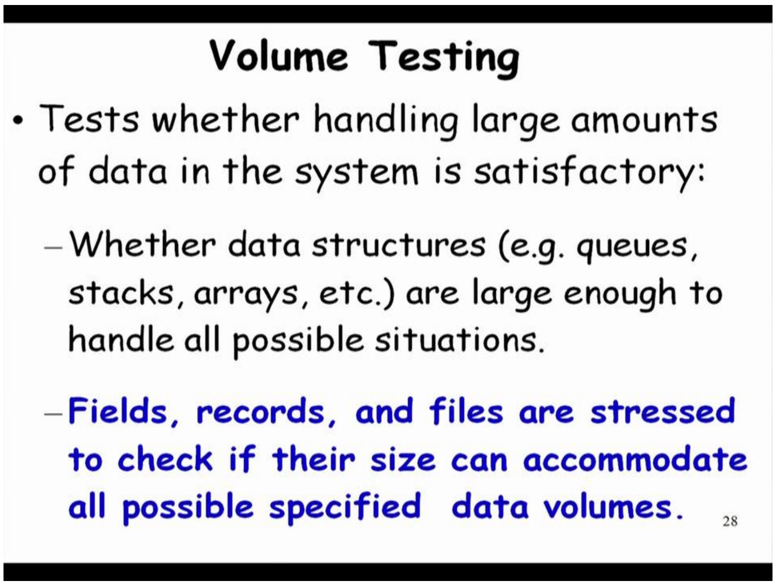
## Load testing

- Load testing:
  - **Determines whether the performance of the system under different loads acceptable?**
  - **Example:** For a web-based application, what is the performance of the system under some specified hits?
- Tool:
  - JMeter: http://jakarta.apache.org/jmeter/

27

Another type of system testing is load testing. In load testing, it is determined whether the performance of the system under different load conditions acceptable. So, there is a clear distinction between stress testing and load testing. In the sense that in stress testing, we give input to the system which is beyond what is specified in the SRS document. Whereas, in load testing we test the performance of the system at different specified loads. Just to give an example, if we have a web-based application and it is specified that the system should, at 100 hits per second, the response times should be less than 1 second. We test here at 90 hits per second, 100 hits per second. So, all within the specified range of inputs, and check is the system behaving as per its requirement. And, there are several load testing tools; one, open source tool, which is very popular is the J meter, which can be downloaded at the apache; jakarta dot apache dot org/ jmeter, which is used for load testing of web-based applications.

(Refer Slide Time: 12:16)



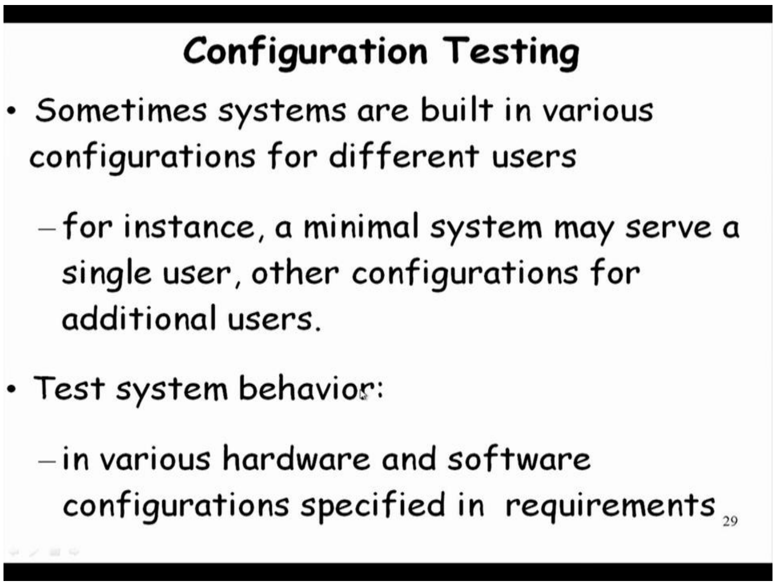
## Volume Testing

- Tests whether handling large amounts of data in the system is satisfactory:
  - Whether data structures (e.g. queues, stacks, arrays, etc.) are large enough to handle all possible situations.
  - **Fields, records, and files are stressed to check if their size can accommodate all possible specified data volumes.**

28

Another performance test is the volume test. In volume test, again the system is tested within the specified bounds as recorded in the SRS document. For example, let us say if the size of the data, that is, queue size is 100, does it handle when there is queue size of 99 or 100? Similarly, the size of arrays, stack, etcetera are they large enough to handle the situations that are possible as mentioned in the SRS document. So, various sizes; for example, fields, records, file size, etcetera. These are stressed to check if they can accommodate all possible specified data volumes. So, this is important here; specified data volumes. So, volume testing; we do not test beyond what is specified data volume. We specify that; we test whether the data volumes as specified in the SRS document are they handled properly by the software?

(Refer Slide Time: 13:50)



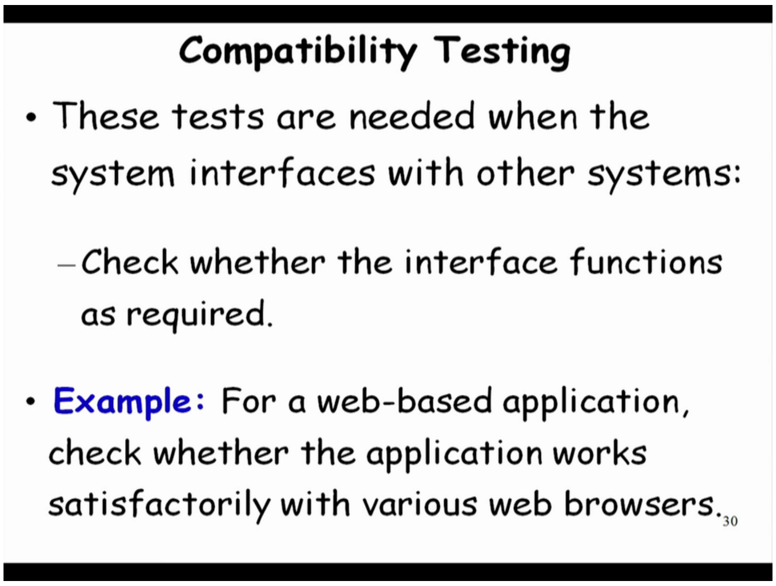
## Configuration Testing

- Sometimes systems are built in various configurations for different users
  - for instance, a minimal system may serve a single user, other configurations for additional users.
- Test system behavior:
  - in various hardware and software configurations specified in requirements

29

Another type of system test is the configuration testing. In configuration testing, which is applicable to software, which is built to support various configurations? For example, we might have a minimal configuration for software, where a single user can use it or we can have a configuration where multiple users can use it or we can have a configuration where it can use certain database and so on. So, the objective of this testing, configuration testing is that does the system work satisfactorily for the various specified configurations of the software.

(Refer Slide Time: 14:44)



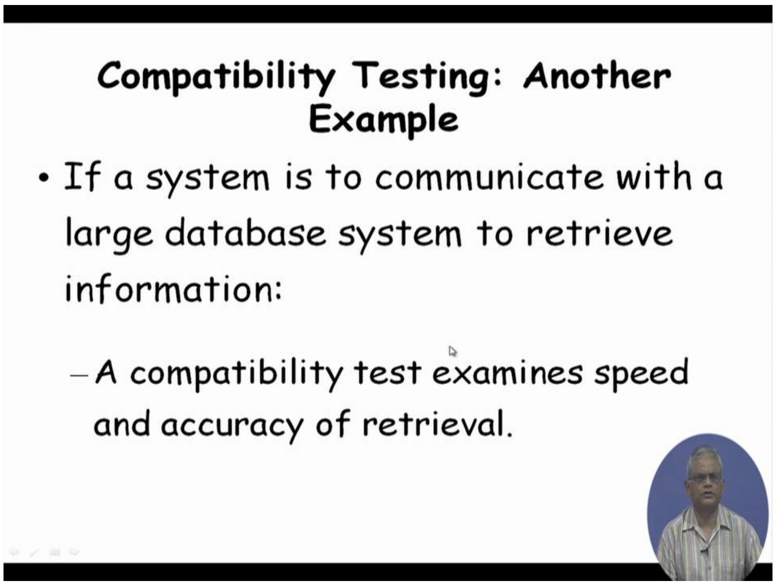
## Compatibility Testing

- These tests are needed when the system interfaces with other systems:
  - Check whether the interface functions as required.
- **Example:** For a web-based application, check whether the application works satisfactorily with various web browsers.

30

Another type of system testing is the compatibility testing. Here the tests are designed to test, whether the software interfaces with other software and hardware as specified in the SRS document satisfactorily. For example, we might have a web-based application, and that is required to work with various web browsers. So, during compatibility testing we check whether the software works with various web browsers such as Mozilla Firefox, the Microsoft Explorer and so on.

(Refer Slide Time: 15:30)

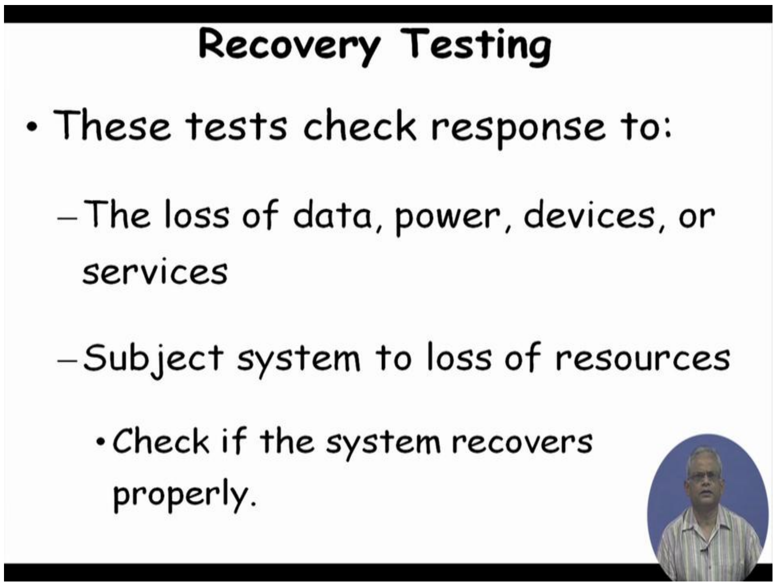


## Compatibility Testing: Another Example

- If a system is to communicate with a large database system to retrieve information:
  - A compatibility test examines speed and accuracy of retrieval.

Another example of a compatibility testing is that if a system has to communicate with a large database system to retrieve information, we check various large databases that needs to be supported. And then, we check the speed and accuracy of retrieval.

(Refer Slide Time: 15:52)



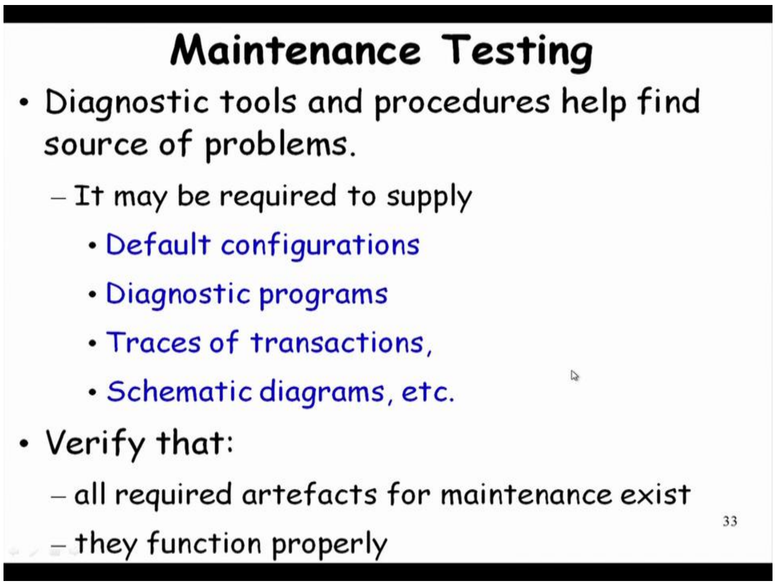
## Recovery Testing

- These tests check response to:
  - The loss of data, power, devices, or services
  - Subject system to loss of resources
    - Check if the system recovers properly.

Another type of system test is the recovery test. So, in recovery test we check whether the software works well, if some loss of power, data, device ( ) etcetera occur. For example, we might be using a pen drive for certain software. And then, suddenly somebody removed the pen drive, so does the software hang or does it say that please put back the pen drive for the software to work?

Or, let us say if the software is mentioned to tolerate power failure, so when power failure occurs, does it lose all the data or does it save the data before the power failure occurs? And then, once the power is given back, does it the systems recover those data properly?

(Refer Slide Time: 16:58)



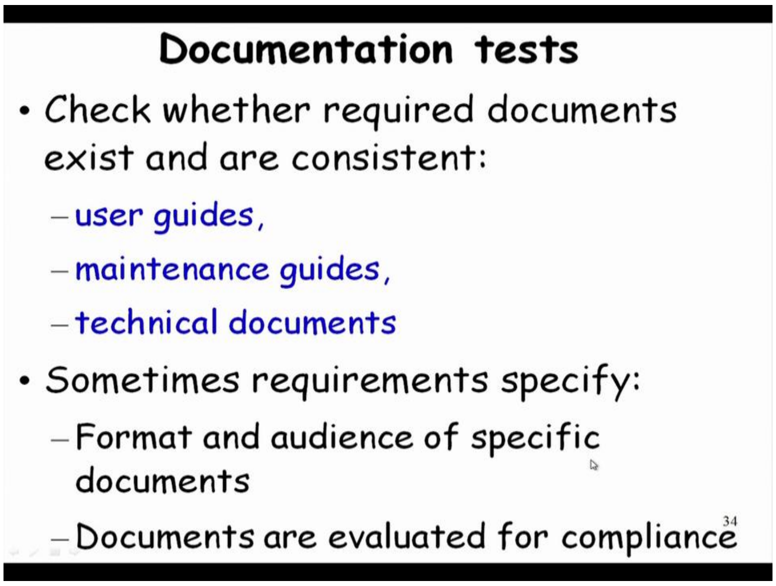
## Maintenance Testing

- Diagnostic tools and procedures help find source of problems.
  - It may be required to supply
    - Default configurations
    - Diagnostic programs
    - Traces of transactions,
    - Schematic diagrams, etc.
- Verify that:
  - all required artefacts for maintenance exist
  - they function properly

33

Another type of system test is the maintenance test. Every software needs various types of maintenance, as it is used over a long time. For example, every software might have some diagnostic programs. If something does not work properly, the speed degrades or certain actions cannot be carried out, then the software typically need to provide default configurations, where somebody can put the software into default configuration. And, it can be checked whether it works properly in the default configuration, diagnostic programmes, and traces of transactions, schematic diagrams about the connectivity to different modules, the way it needs to be connected to other software and hardware and so on. So, these are various maintenance testing it keeps. And, whether these have been supplied and whether these function properly?

(Refer Slide Time: 18:11)



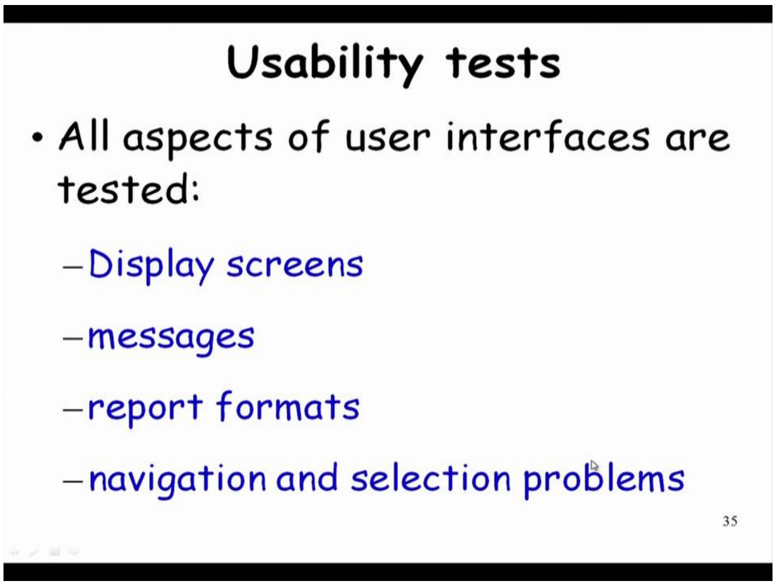
## Documentation tests

- Check whether required documents exist and are consistent:
  - user guides,
  - maintenance guides,
  - technical documents
- Sometimes requirements specify:
  - Format and audience of specific documents
  - Documents are evaluated for compliance

34

One more type of system test is the documentation test. So, here in this type of test we just need to check whether all the provided documents are consistent and they have been provided; because the user when doing testing, acceptance testing, will also have to do the documentation test, where the user need to check whether the user guides, maintenance guides, technical documents, etcetera, they have been provided are consistent. Sometimes for some software, it may be required that they support various types of users. For example, very novice users, experienced users and so on. And, whether the documents that have been provided for those category of users are as per requirement of those types of users.

(Refer Slide Time: 19:16)



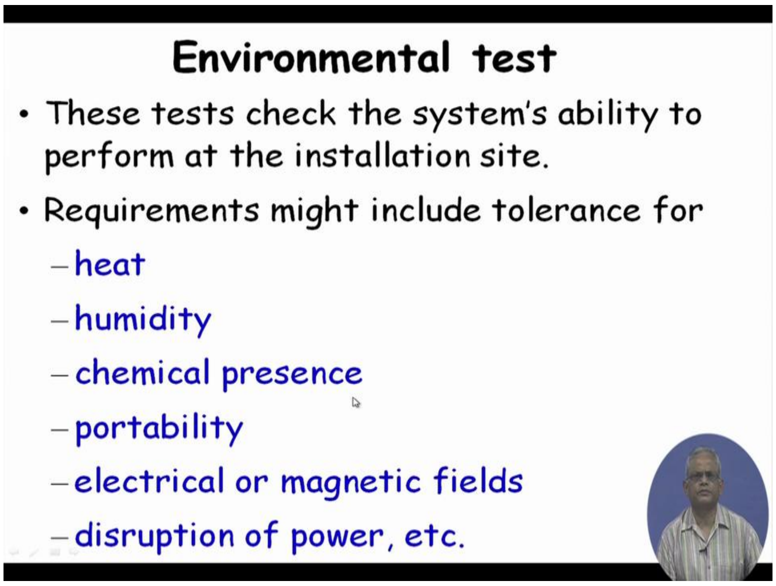
## Usability tests

- All aspects of user interfaces are tested:
  - Display screens
  - messages
  - report formats
  - navigation and selection problems

35

One more type of system testing is the usability testing. So, here the target is to test the user interface; various display screens, messages, report formats and so on. They are checked whether they are okay, and also navigation in the user interfaces; are it, is it satisfactorily working, menu selections and so on. Mouse clicks; are these properly recorded and handled.

(Refer Slide Time: 19:54)

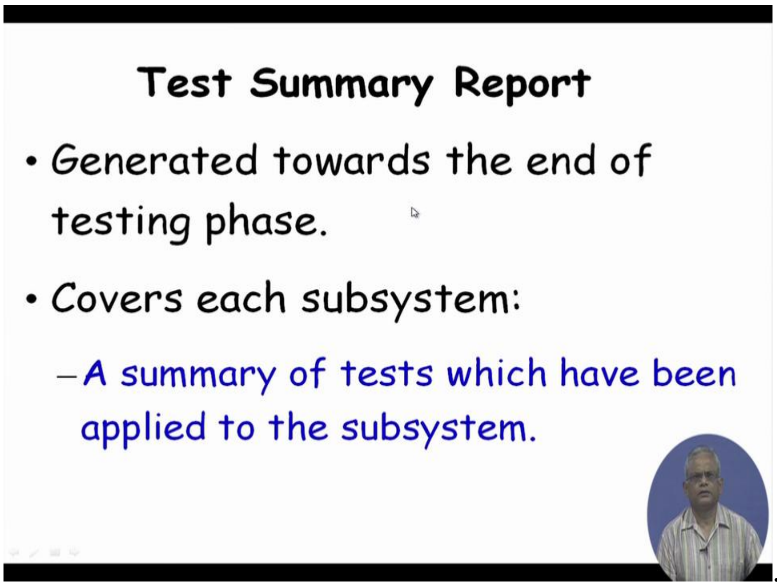


## Environmental test

- These tests check the system's ability to perform at the installation site.
- Requirements might include tolerance for
  - heat
  - humidity
  - chemical presence
  - portability
  - electrical or magnetic fields
  - disruption of power, etc.

One more type of testing, which is environmental test; which is not applicable to general software. Possibly, applicable to software that is required to work in devices which might work in different climatic conditions. For example, embedded systems. So, here it is checked whether the software which is may be controlling an embedded device, does it work on the different heat, humidity, chemical presence, portability conditions, electric, magnetic fields, disruption of power, etcetera.

(Refer Slide Time: 20:35)



## Test Summary Report

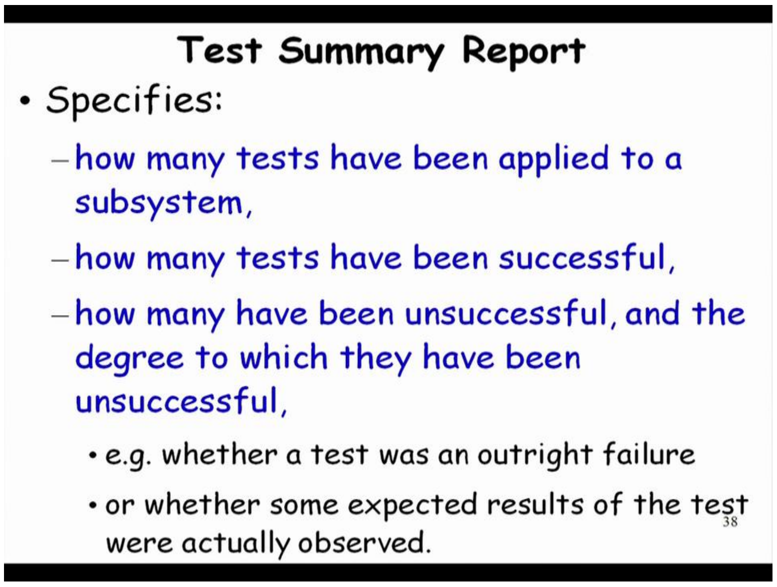
- Generated towards the end of testing phase.
- Covers each subsystem:
  - A summary of tests which have been applied to the subsystem.

.

Now, once all the tests have been done, the system tests. So, that is the test carried out either by the development team or a friendly set of teams, which is the beta testing or the customer himself, which is the acceptance testing.

The test observations are to be written down in a test summary report. So, what do we write in the test summary report? In the test summary report, we have to write the summary of all tests, which has been applied to the test that to the software.

(Refer Slide Time: 21:16)



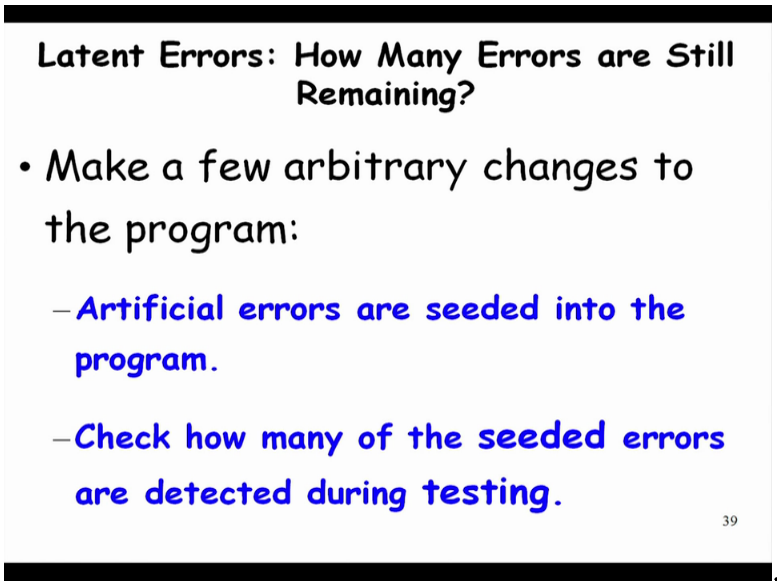
## Test Summary Report

- Specifies:
  - how many tests have been applied to a subsystem,
  - how many tests have been successful,
  - how many have been unsuccessful, and the degree to which they have been unsuccessful,
    - e.g. whether a test was an outright failure
    - or whether some expected results of the test were actually observed.

38

So, both functional testing and the performance testing. How many tests were applied, how many tests were successful, how many tests have been unsuccessful and the degree to which they were unsuccessful. Was it that in the usability test, in some fields, you have to the click the mouse twice. So, that is a minor problem or is it that it crashed. So, whether the test was an outright failure or whether some minor unexpected results, etcetera were only observed. So, that should be the contents of the test summary report.

(Refer Slide Time: 22:00)



## Latent Errors: How Many Errors are Still Remaining?

- Make a few arbitrary changes to the program:
  - Artificial errors are seeded into the program.
  - Check how many of the seeded errors are detected during testing.

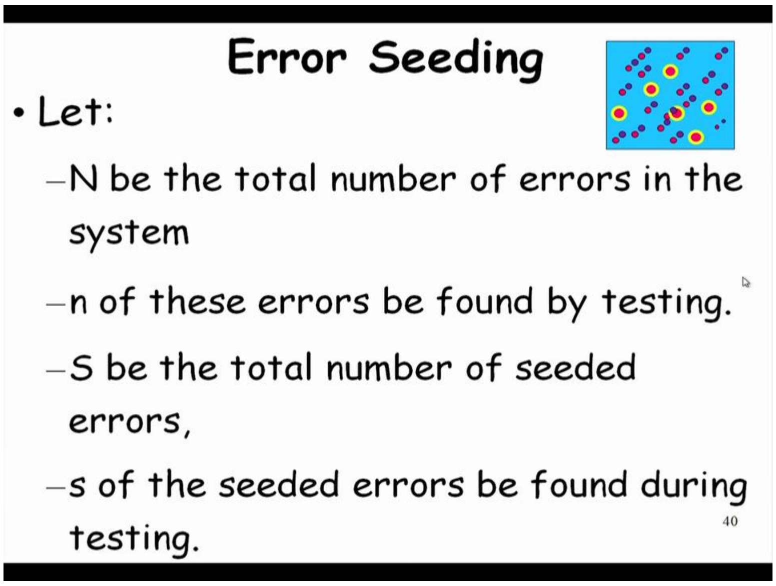
39

.

Now, let us look at one problem which bothers every developer, every tester and every user of software; that how many latent bugs are there in the software. So the software let us say a million line of code has been developed. After a lot of work by the developers, it has been tested. Now, if somebody asks that what is the likely number of bugs or faults that are existing in the software, after all these careful development and testing has been carried out, how do we give a number, how does the project manager determine that how many bugs are still there?

So, one way to determine it is called as error seeding. So, how do we do the error seeding and how does it help us determine the number of bugs in the software? Here, the main idea is that before the testing starts, the project manager or a senior person introduces some bugs artificially into the software. So, makes a few changes in the software, which he keeps track off. He has recorded what type of bugs he has introduced. May be, in some places he has flipped an arithmetic operator, plus operator into a minus operator or somewhere in the conditional statement, might have made a less than equal to into a less than operator or may be somewhere where it is written a + b + c, he might have written only a + b and deleted c and so on. So, these are some of the bugs that the manager seeds. And then, at the end of the testing he finds out how many of these seeded bugs have been discovered. If he seeded hundred bugs, out of the hundred bugs, how many of his seeded bugs could be discovered?

(Refer Slide Time: 24:24)



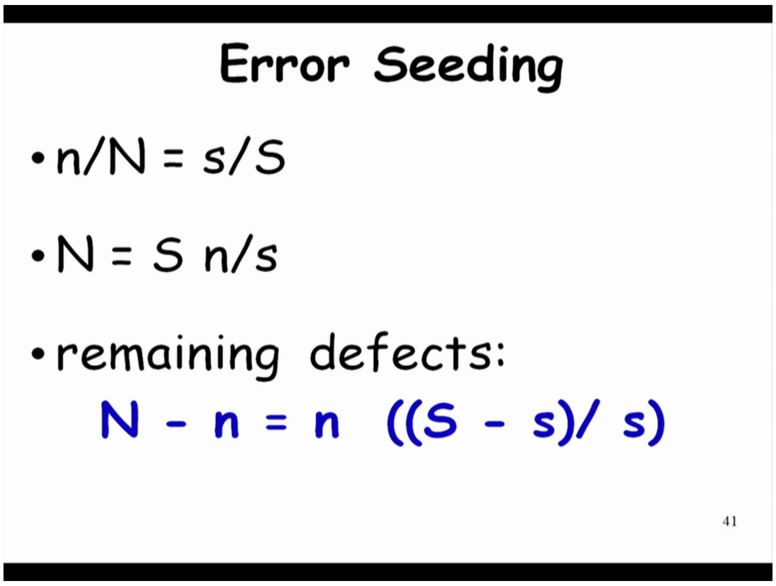
## Error Seeding

- Let:
  - N be the total number of errors in the system
  - n of these errors be found by testing.
  - S be the total number of seeded errors,
  - s of the seeded errors be found during testing.

40

And, so let us say these are the bugs which are existing in the software. And then, the manager seeded some bugs in the software. So, the initial number of bugs which existed let us say this is capital N. And, now the manager seeded a set of bugs there; made arbitrary changes. And, small; and s is the seeded bugs. So, s number of bugs he seeded. And then, the testing was carried out. And at the end of testing, it was found that it has; small s numbers of seeded bugs were discovered during testing and small n number of bugs, which existed, they have also been discovered by testing. So, these are the bugs which have been discovered; the seeded bugs and the unseeded bugs.

(Refer Slide Time: 25:35)



## Error Seeding

- $n/N = s/S$
- $N = S\ n/s$
- remaining defects:
  $N - n = n\ ((S - s)/ s)$

41

Then, we make a simple assumption that if small n out of existing number of bugs is detected that must be equal to s / S. So, we are claiming here or we are assuming here that the testing identifies the bugs in the same proportion to both the regional bugs and the seeded bugs. And, from this expression we get the remaining bugs $N - n = n * ((S - s) / s)$. So, this gives us a very rough approximation of the number of bugs that are adjusting in the software. So, error seeding is an important technique to find out what is the number of bugs that has been left behind in the software, even after doing a careful development and carrying out thorough testing. So, we will stop this session at this point and continue from this point onwards in the next session.

Thank you.