Software testing

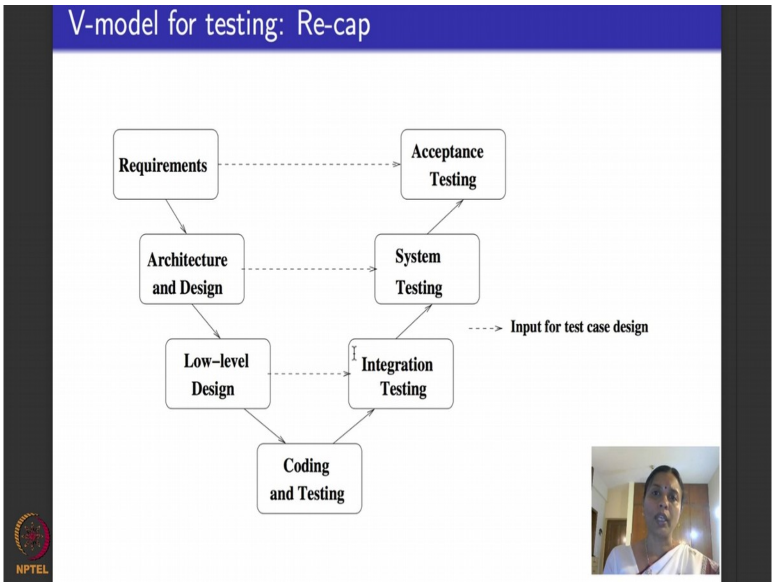
Prof. Meenakshi D'Souza
Department of Computer Science and Engineering
International Institute of Information Technology, Bangalore

Lecture – 57
Non-functional System Testing

Hello again, welcome to the last weeks next lecture. Again as I told you during the mobile apps testing lecture, in response to my pinging on the forum about what you would like to have, a few of you had also requested for a module on performance testing. So, here is a module that caters to that.

This particular lecture does not do only performance testing because either you do it by opening a tool, some of them proprietary and cater to a web application or you give an overview. If I give an overview, then it would not be enough for one lecture, but if I open a tool and then goes out of the focus of this course, which was focusing on algorithms. So, what I decided I will do is that I will give you on a overview of all nonfunctional, popular nonfunctional system testing techniques and performance testing will come as a list in this set of techniques. So, that is what I thought I will cover. Next module I will do regression testing. So, here is an overview what we are going to do in this lecture.

(Refer Slide Time: 01:38)

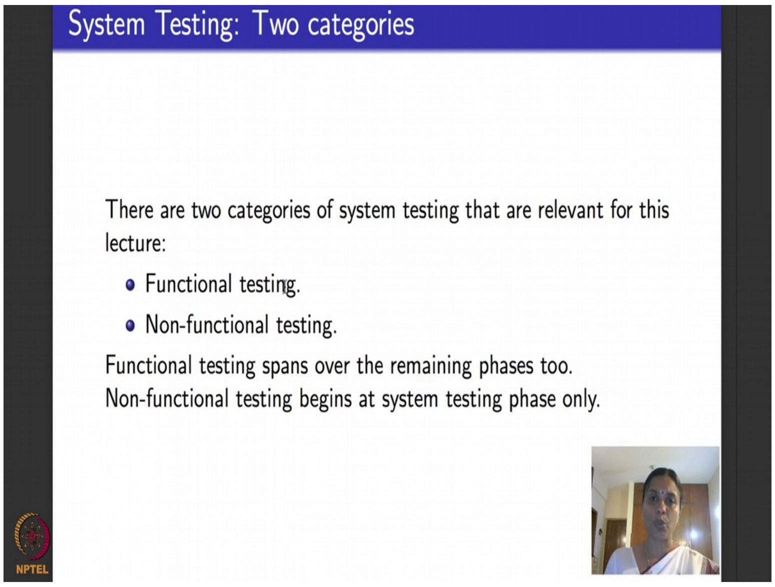


I will give you a taxonomy of system testing that is typically done and we will focus on as I told you non-functional system testing and as requested, I will also look into a little more details about performance testing by giving you tools that are open source proprietary, what are the goals to measure and so on. So, if you remember, I had shown you this V model long ago when we began in the first week in the course. So, on the left hand side of the V model is what we have, the normal software development life cycle. We do requirements architectural design, coding, low level design, coding and then we do testing. But when it comes to focus testing, we usually expand the V model into its right arm which tells you what is the kind of testing that is done.

Right here at the bottom, along with coding, I do unit testing. In an agile methodology typically done by the developer, after which I do integration testing, software modules are an integrated, software is integrated with hardware database and tested. Then we do system level testing where everything is put together with the entire working system in test it finally, acceptance testing. This model we had looked at long ago. What did we focus on in the course, till the past few weeks, they were here and here, coding unit testing and integration testing. What is be in the focus of this week and in the past when we did web applications and object oriented testing, that has been system level testing. Today again will look at system level testing, but not focus on functional system level testing instead focus on what is called non-functional system level testing.

Again in the first week in my introductory lectures, I had introduced you to 2 kinds of testing, several dichotomies testing and one of it was functional and non-functional.

(Refer Slide Time: 03:14)



System Testing: Two categories

There are two categories of system testing that are relevant for this lecture:

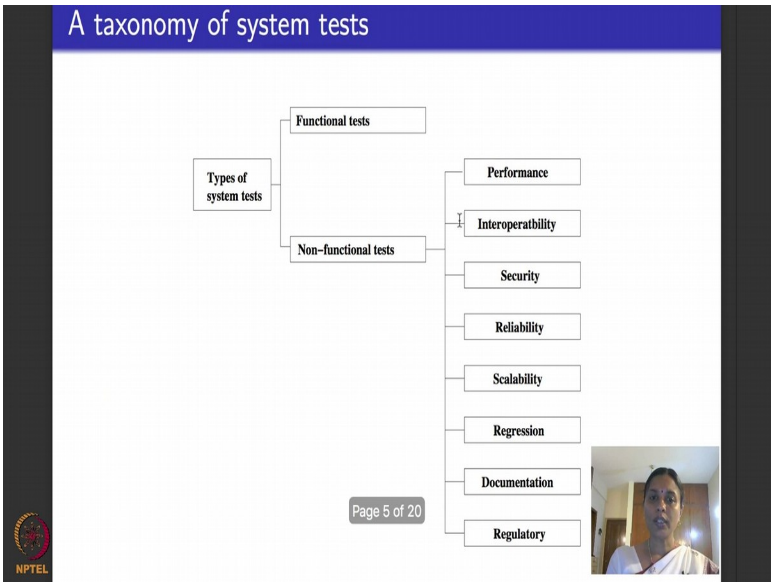
- Functional testing.
- Non-functional testing.

Functional testing spans over the remaining phases too.
Non-functional testing begins at system testing phase only.

So, here is the category that we did, right in the first week that we are recapping again, there are 2 category of system level testing that we would like to recap. One is called functional testing which we have looked out majority of the course, basically tests if the software needs its requirement, white box, black box.

Anyway, the next is called non-functional testing which basically tests the software to check if it needs various quality attributes. As per ISO standard, there are 64 different quality attributes most of them ending with the word "LITY" or TY. So, many organizations popularly abbreviate them and call them as "ilites" which is what nonfunctional testing focuses on. Non-functional testing typically begins only at the system level. We do not really do non-functional testing, here at coding and unit testing or at integration testing, but the work for non-functional testing, as we saw in the V model begins along with requirements architecture and design. Typically, it is the architecture that caters to quality parameters the work in terms of what to test for, input for text case design, they get ready here actual testing happens here.

(Refer Slide Time: 04:29)



A taxonomy of system tests

$$\begin{tikzcd}[row sep=small]
 & \text{Functional tests} & \\
\text{Types of system tests} \arrow[ur] \arrow[dr] & & \text{Performance} \\
 & \text{Non-functional tests} \arrow[ur] \arrow[r] \arrow[dr] & \text{Interoperatbility} \\
 & & \text{Security}
\end{tikzcd}$$

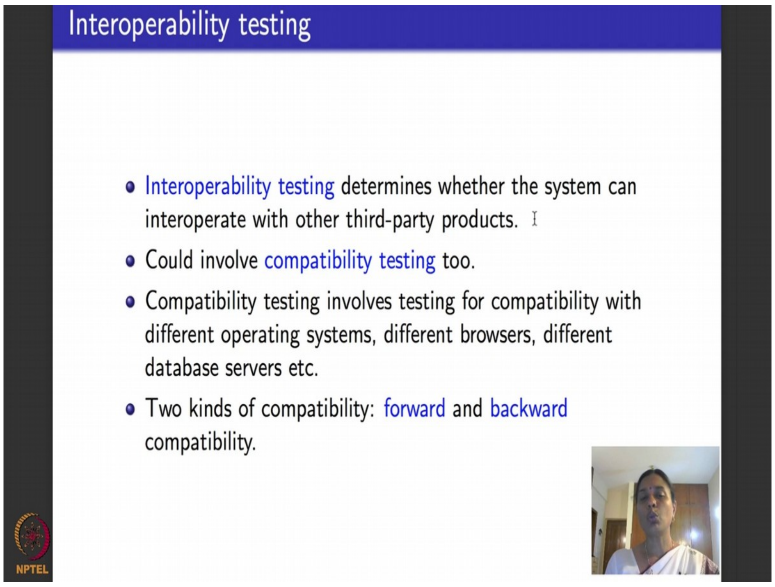
- Performance
- Interoperatbility
- Security
- Reliability
- Scalability
- Regression
- Documentation
- Regulatory

Page 5 of 20

So, our focus is on nonfunctional testing. So, here is the same dichotomy which expands into nonfunctional tests. So far, our classification, there are 2 kinds of system tests: functional system test and nonfunctional system test. There is several different varieties of nonfunctional system test, I have chosen to put a few popular among them and I should also explicitly tell you that I have left out a good number of them. So, this by no means is an exhaustive listing of all the non-functional tests.

What we will be doing in this lecture is sort of go through them one after the other. So, performance, I will reserve for the end. We will begin with interoperability, follow it with a brief overview on security and reliability. To iterate what I told you in the context of mobile apps testing, this to a very involved area. This course what we do is by no means even a cursory introduction to these elaborate testing fields, we look at what is scalability testing for scalability regression testing, I will do a separate half an hour module because again, there was a request for that today I will also tell you about documentation and regulator testing. So, system test 2 categories functional and nonfunctional are focused today is nonfunctional test and specifically re-look at all this except for regression testing which will do in a separate module, and security and reliability need in depth analysis which we will not be able to do in the course due to lack of time. So, rest I will tell you. We begin with interoperability testing.

(Refer Slide Time: 06:03)



Interoperability testing

- Interoperability testing determines whether the system can interoperate with other third-party products.
- Could involve compatibility testing too.
- Compatibility testing involves testing for compatibility with different operating systems, different browsers, different database servers etc.
- Two kinds of compatibility: forward and backward compatibility.

So, what is the word interoperability tell you or remind you of? It says you something should work with the other across platforms; across operating systems, across languages. So, that is what interoperability testing does. It determines whether the system can operate or interoperate along with other third party products.

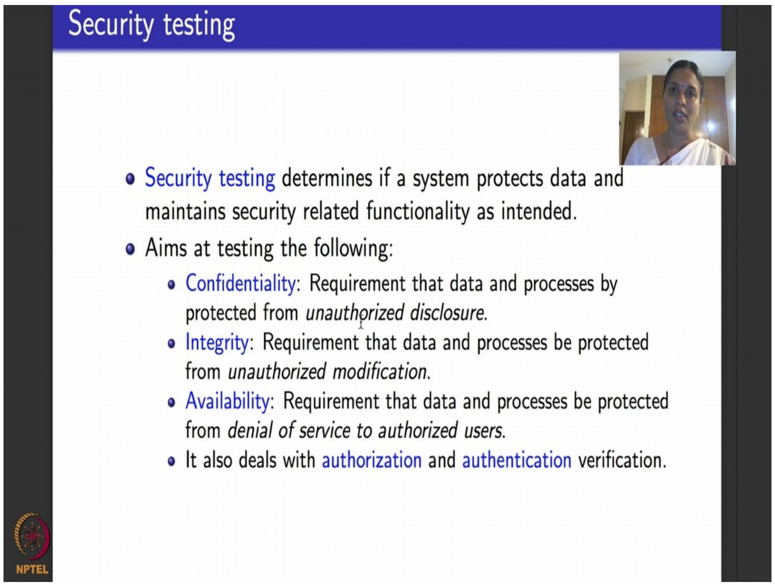
Suppose I say that I have recording software and it should be able to operate with any kind of camera on any system. It could be installed for a MAC system for a windows system, or for a Linux system, but whatever it is, let us say there is there is an inbuilt camera, there is an externally fixed camera there is a USB camera port whatever it is my recording software should be able to pick up the data from the camera and record. So, it should be interoperable along with the third party software or a hardware component. That is what interoperability testing does. A particular wing interoperability testing is what is called compatibility testing.

What is compatibility testing do? It involves checking for compatibility with different operating systems, different browsers, different database servers and so on. When we did the module on mobile apps testing, I had spoken to you a bit about this. So, when I say I have an app, a good enough app, I make it amenable to run on an Android OS or an IOS and so on. Similarly I have piece of something, let us say a tool like Junit, will it run on Windows machine, will it run on Linux machine, that is what compatibility testing does.

Compatibility testing also checks for forward and backward compatibility. Let me you give an example for that let us take you, let us say you have Microsoft office software which has Microsoft office tools suite, power point, Microsoft office word, excel, outlook. So, many other things right and they keep releasing newer and newer versions. So, now, let us say you have an office word document that you open with an older version, let us say 2007, it should be compatible with the newer version which ever, in the sense that if I had developed document with an older version, I should be able to open and edit it in a newer version. That is what is call forward compatibility.

Backward compatibility means something was developed in a later version then I should be able to at least open it in the previous version, may not be able to edit it if it uses features of newer features of the later version, but I should at least be able to open it and view it. So, interoperability also deals with checking for compatibility. Sometimes you will get messages, especially when you are working with some websites operated by governments banks saying that this is compatible only with the following browsers and the following versions of these browsers which means what it has features that suit only these browsers and these versions. It is not compatible with other browsers and their versions that are not specified in the list. So, this is another important non-functional testing that people do at a system level testing for software or a system.

(Refer Slide Time: 09:29)



Security testing

- Security testing determines if a system protects data and maintains security related functionality as intended.
- Aims at testing the following:
  - Confidentiality: Requirement that data and processes by protected from *unauthorized disclosure.*
  - Integrity: Requirement that data and processes be protected from *unauthorized modification.*
  - Availability: Requirement that data and processes be protected from *denial of service to authorized users.*
  - It also deals with authorization and authentication verification.

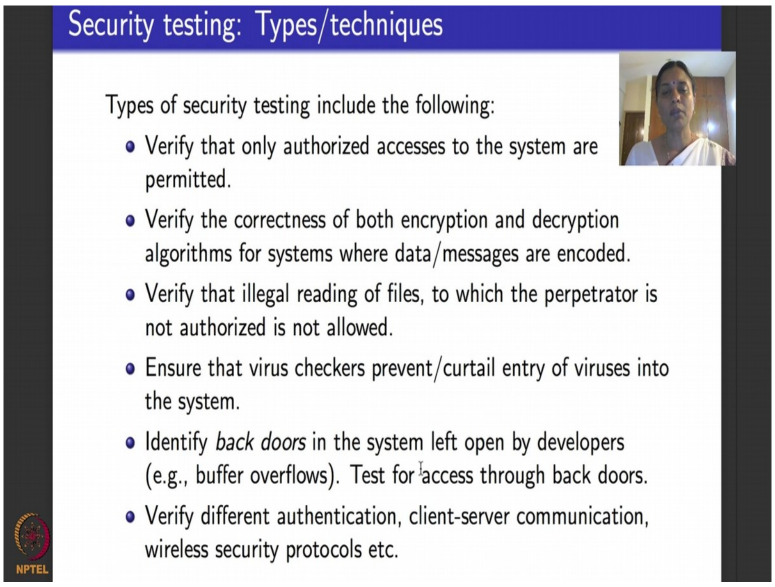
So, the next security testing. Just to repeat that the cost of being repetitive, I had tell you that we are only scratching the surface, this is very involved area feel free to browse through other courses of books that talk about it, I am just giving you the introduction to that term. So, it determines if a system or a product protects the data it handles with lot of data like banks means our account data that sensitive and private to us, user name password and so on, and it does it maintained security related functionality as it was intended to be. Security testing involves further "ilities": confidentiality, integrity and availability.

What is confidentiality; say as the word indicates, it is a requirement, it tests for requirement that data and processes set up by a software or a system, is protected from unauthorized disclosure. I get my bank information provided I have specified the credentials correctly. I get my system marks on-line provided, it is me, I have provided the user name and password correctly; that is confidentiality. Integrity is tests for requirement the data and processes be protected from unauthorized modification. Somebody manages to get access to my data, person is an authorized to access my data then the person continues to be unauthorized to modify my data because that is even bigger than accessing the data, that is integrity testing.

The next one is availability testing. It tests that data and processes be protected from denial of service to authorized users, means if a particular authorized person is authorized to have access to a particular data or a service, then the data or the service is made available to user. So, that is what availability testing. So, security testing tests for confidentiality, integrity, availability, it also deals with authorization and authentication validation. Authorization to give you an example, typically you might say that if something is available in the private mode then only the person who owns it is authorized to view it, but something is made public then everybody is authorized to view it. So, it gives permissions to save who is authorized to view access read or write to certain content.

Authentication is the process of validating a particular user. There are authorization policy is like role based authorization, MAC, DAC and so on that are popular across system software provided by several different products and typically security testing involves validating for authorization and authentication policies also.

(Refer Slide Time: 12:16)



Security testing: Types/techniques

Types of security testing include the following:

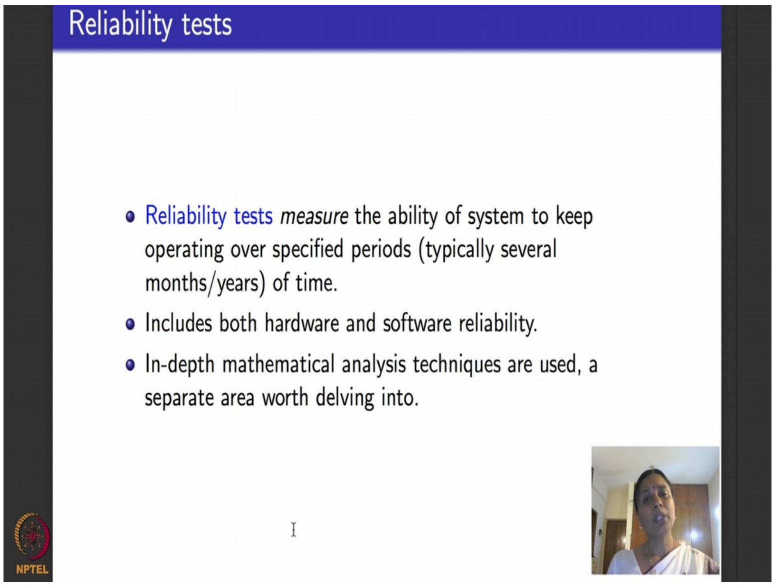
- Verify that only authorized accesses to the system are permitted.
- Verify the correctness of both encryption and decryption algorithms for systems where data/messages are encoded.
- Verify that illegal reading of files, to which the perpetrator is not authorized is not allowed.
- Ensure that virus checkers prevent/curtail entry of viruses into the system.
- Identify *back doors* in the system left open by developers (e.g., buffer overflows). Test for access through back doors.
- Verify different authentication, client-server communication, wireless security protocols etc.

So, the various types of security testing will test for the following. We will verify that only authorized access to the system are permitted. It will also verify that the encryption and decryption algorithms that are used for encoding data or messages work correctly. It also verifies that illegal reading of files to which a particular user is not authorized is actually not allowed. It ensures that if you have an anti-virus software, they actually prevent, far from the preventing they also curtail entry of viruses into the system.

Typically security testing, also testing tools and techniques also identify what are called back doors in the system.

Back doors in the system referred to classical coding mistakes that developers could make which provide illegal access to an unauthorized user or hacker. One of the back doors is a buffer overflow which hackers supposedly user to use to hack into a system. So, security testing also systematically works towards identifying such back doors and ensures that access through these back doors actually not possible. As I told you, it verifies for authentication, authorization and there are several protocols at various layers that these security policies use. Based on what their communication interfaces, they could use several wireless security protocols, client server security protocols, authentication protocols. Each of them has to be separately tested at appropriate different layers security testing covers for all of them also.

(Refer Slide Time: 13:57)

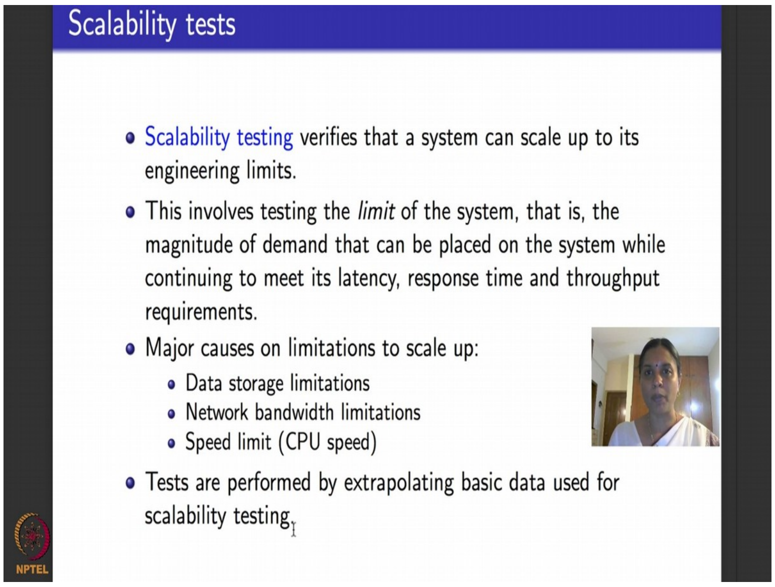


Reliability tests

- Reliability tests *measure* the ability of system to keep operating over specified periods (typically several months/years) of time.
- Includes both hardware and software reliability.
- In-depth mathematical analysis techniques are used, a separate area worth delving into.

Next is reliability. The term reliability indicates that the system is reliable in the sense that if it runs for really really long time, it continues to perform, does not degrade. Let us say I buy a particular electrical appliance, what does it mean for an electrical appliance to be reliable? Let us say if I buy a fan, what I mean it to be reliable; I expect it to work for many years without failing, that is what I mean for a system to be reliable. So, what is reliability testing do its test to measure the ability of a system or software to keep operating over a specific period of time. This specific period of time is usually long measured in terms of years or sometime months. Reliability includes hardware reliability and software reliability. It involves different techniques. Both of them deploy a lot of probability theory and Statistics which we have not covered in this course. So, this is a separate area that you can run a separate course on, there that much of information and literature available.

(Refer Slide Time: 14:58)



Scalability tests

- Scalability testing verifies that a system can scale up to its engineering limits.
- This involves testing the *limit* of the system, that is, the magnitude of demand that can be placed on the system while continuing to meet its latency, response time and throughput requirements.
- Major causes on limitations to scale up:
  - Data storage limitations
  - Network bandwidth limitations
  - Speed limit (CPU speed)
- Tests are performed by extrapolating basic data used for scalability testing.

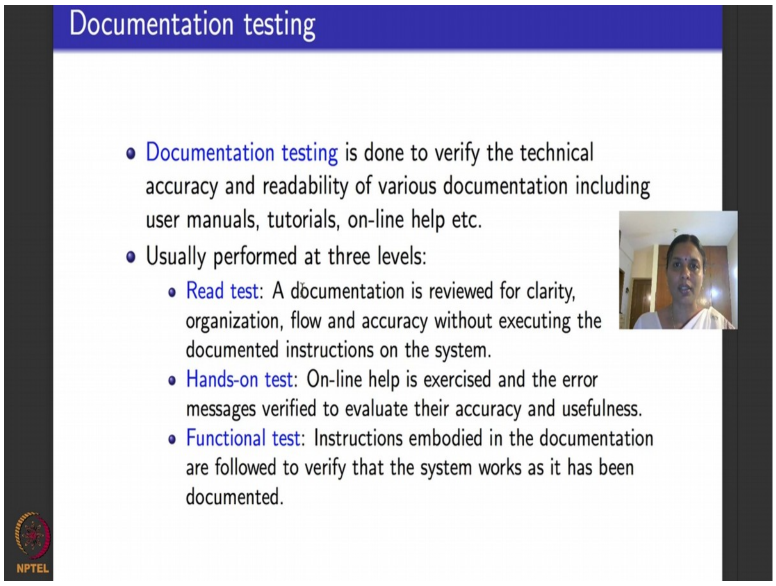
So, the next in a list of nonfunctional system level testing would be scalability tests. What does it do? It says will the system scale in the sense that let us say a particular server or a online web software lets say HDFC bank online, it is supposed to cater to 100-1000 simultaneous users logging in and accessing the respective account information. Another classical example is our Indian railways booking website, right, is it scalable. The normal load could be several 1000 users during Tatkal, it could be get higher number. Does it cater to the maximum number of users that it is supposed to cater to? Can its scale up to its limits? The limits need not be only in terms of users, it could be in terms of any engineering parameter that we specify. For example, one is the number of users, the other is the size of the database that it can manipulate, like for example, if you take an Aadhar system, right, Aadhar system should be scalable to cater to our country's population. So, that is it engineering limit, right.

So, scalability testing involves testing the limit of a system that is it is the magnitude of demand that can be placed on the system as it continues to perform. Perform in the sense that it has a committed response time, no matter if there are ten users logged into the Indian railways ticket booking website or if there are 100 users or 1000 users logging into Indian railways ticket booking website, it still meets its commitment of booking tickets upon availability within a specified time. So, that is what scalability testing does. So, what could be the causes that can limit a system from being not being scalable. Typically, it could the typical major causes that prevents the system from being scalable are, the data storage could be limited, in the sense of accessing the database and actually storing the data. It is a very big problem actually to be able to store data in such a way that it get it does not get heated up, especially for large systems like Facebook and Whatsapp and all data storage is a huge problem or even Google and any things like that.

The other big problem that could limit scalability is network bandwidth. It could be less, it should work, it should switch over to non GUI graphics, plain html kind of thing. Next limiting factor could be the speed limit, by speed limit, I actually here mean the CPU speed; the processor speed. So, scalability tests, actually what they do is that it is very difficult to create use case for a maximum number of users. Let us say a system says it can cater to up to 500000 users, it is very difficult to actually write a test case that will create a scenario of 500000 users logging to a system and accessing data. Such kind of test cases are very expensive to create.

So, what they do is that they create test cases to the extent that they can and then they extrapolate the basic data and test the system for scalability, that is how scalability testing works. So, it again needs a lot of statistics.

(Refer Slide Time: 18:15)


Documentation testing

- Documentation testing is done to verify the technical accuracy and readability of various documentation including user manuals, tutorials, on-line help etc.
- Usually performed at three levels:
  - Read test: A documentation is reviewed for clarity, organization, flow and accuracy without executing the documented instructions on the system.
  - Hands-on test: On-line help is exercised and the error messages verified to evaluate their accuracy and usefulness.
  - Functional test: Instructions embodied in the documentation are followed to verify that the system works as it has been documented.

So, the next kind of non functional system testing that was there in my list is documentation testing. So, basically whenever you have a software or a product or a system, it comes with several documentations, like a user manual which could be an online manual or a printed manual, it could have tutorials, it could have an online help, lot of these kind of things are there. So, they constitute what is called the documentation of software, apart from the actual code also being documented.

So, documentation testing is basically checking if all this documentation is correctly done to the extent that it is supposed to be. So, it is done to verify the technical accuracy, it is correct or wrong technically and the readability, is it readable, is it understandable, of the various documentations. Such kind of documentation includes user manuals tutorials online help and so on. Documentation testing is usually performed at 3 different levels the first level is what is called a read test.

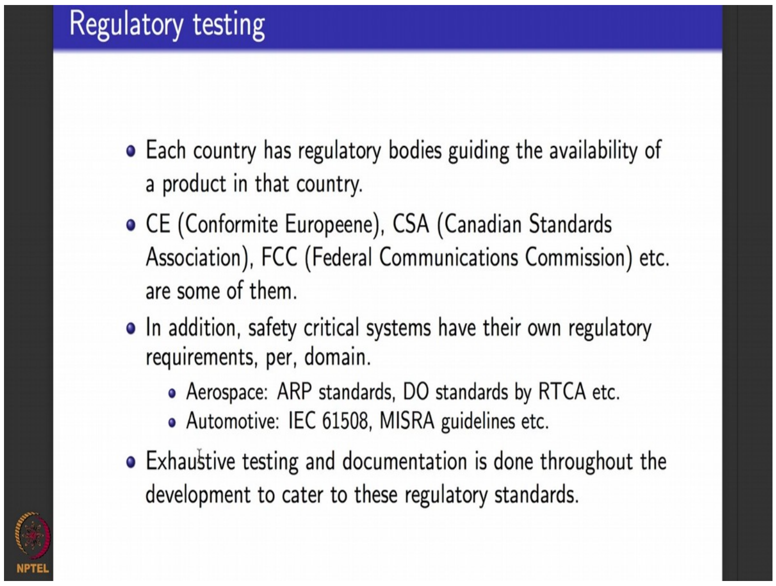
Basically somebody sits, competent in the area, sits and reads the documentation. In that process, the documentation is reviewed for clarity, for organization in the sense, the contents, are they laid out correctly, a chapter is organized properly, sections are organized properly. The flow and the accuracy, this is just static documentation testing nothing is executed. Then there is something called hands on documentation testing where an online help is actually exercised by a user, like a dummy login is created and then they go as if they do not know anything, look for online help, see if it provides the right kind of help, see if it provides the right kind of error messages and then they sit and evaluate how useful it is, how accurate it is.

Then there is something called functional documentation test which basically checks of the instructions that are given in the documentation are followed and verifies so that the system works as it is supposed to be. It means its functionality as documented. So, some of the tests that people usually do while doing documentation testing are as follows. Somebody as I told you manually reads through all the documentation to check if the grammar is correct, there are no typographical errors, technical terms has been consistently used and so on. Somebody reviews the use of graphics images like for example, you know typically, let us say you are buying a product like a car, then there could be several views of a car, in certain view you can see certain parts of a car.

So, it is the graphics giving the correct view of a car, correct view of the cross section of a car, of the front of a car, of the chassis of a car and so on, that is what people check for graphics and images. Then somebody actually verifies the glossary of terms that a company, the documentation and somebody verifies that there is a proper indexing of terms that is available as a part of the user manual. For every index, as you typically see, there will be a term and a page number where the term appears. So, they go and manually check if the page numbers have been correctly generated. Most documentation generated, software can do this automatically and the scope for error is less because it is automatically done.

But people typically still go and check. This is a part of their routine documentation checks. Then they go and verify that the if you have a printed version and an online version of the user manual they are actually one and the same; they are not different, they do not differ from each other, then they verify if the specified installation procedure is actually working correct by executing the steps on the actual system, then they verify the troubleshooting guide with the actual scenarios. So, this is all, it is quite exhaustive, they do a lot of exhaustive documentation testing and it is a pretty non trivial non-functional testing.

(Refer Slide Time: 22:15)


Regulatory testing

- Each country has regulatory bodies guiding the availability of a product in that country.
- CE (Conformite Europeene), CSA (Canadian Standards Association), FCC (Federal Communications Commission) etc. are some of them.
- In addition, safety critical systems have their own regulatory requirements, per, domain.
  - Aerospace: ARP standards, DO standards by RTCA etc.
  - Automotive: IEC 61508, MISRA guidelines etc.
- Exhaustive testing and documentation is done throughout the development to cater to these regulatory standards.

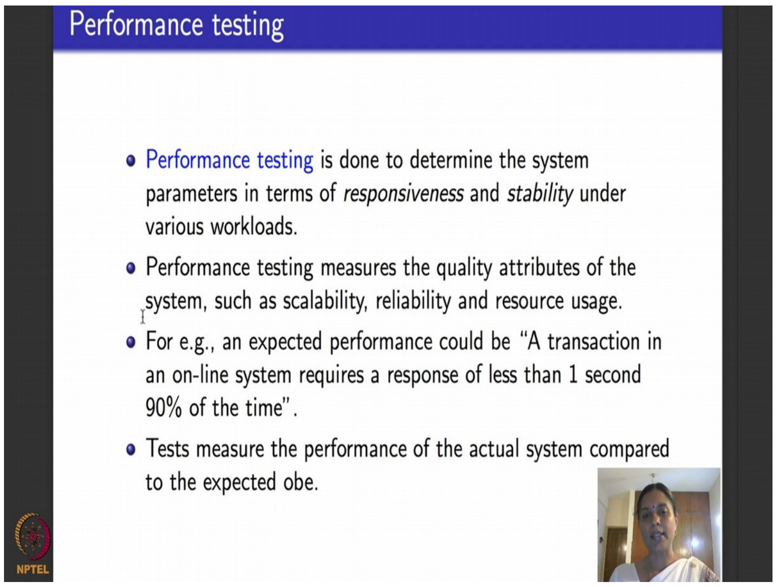
So, the next in a list is regulatory testing. What do we mean by regulatory testing? If you are bought a toy, you know typically you would see that it has these words called CE printed in a little sans serif style. So, that stands for Conformity European. Sometimes you would see FCCs, these days you see FSSAI in food products. So, these are all regulatory standards that describe or tell you that the particular product including software operating on a system, is of a certain quality. So, to be able to get that kind of regulatory assessment passed, usually software and system developers have to specifically undertake a testing called regulatory testing. They basically insist that you do a huge amount of documentation and perform routine tests as mandated by the corresponding regulatory authority.

In addition to this, if you have safety critical systems, like for example, the drive by wire, brake by wire, steer by wire or the power window controller and a car, the ABS system, the airbag system or let us say autopilot of a airplane, the landing system of an airplane, these are safety critical because if they fail then they mean huge losses in terms of loss of life.

So, safety critical systems have to go through extra regulatory requirements. I had told you about this on and off in my lectures about this, There is this particular standard called DO standards for avionics. Similarly for automotive there is IEC 61508, Misra guidelines and so on. In fact, latest you might have heard of this Google and Tesla's driverless cars. So, they have even stricter regulatory standard. So, basically people do regulatory testing to cater to a particular standardization based on the domain of applicability of the software and the system.

So, now we finally, get into performance testing. So, what is performance testing do? It is testing done to determine if the system parameters in terms of responsiveness and stability work fine under various workloads.

(Refer Slide Time: 24:15)



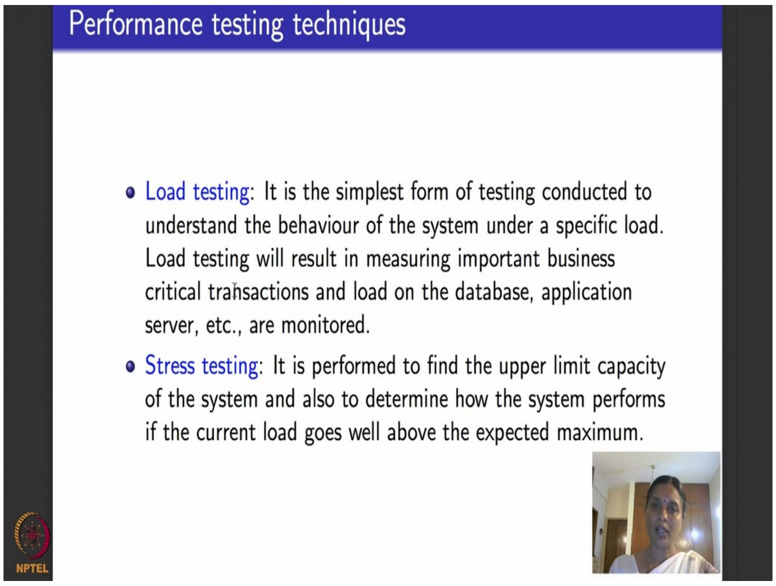
Performance testing

- Performance testing is done to determine the system parameters in terms of *responsiveness* and *stability* under various workloads.
- Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.
- For e.g., an expected performance could be "A transaction in an on-line system requires a response of less than 1 second 90% of the time".
- Tests measure the performance of the actual system compared to the expected obe.

So, performance testing measures the performance of the system for several different quality attributes. Typically it performs it for scalability, reliability, resource usage and a few others that we will see very soon. So, for example, here is a typical example of how a performance testing requirement could be. A transaction in a particular online system requires a response of less than one second, ninety percent of the time.

So, what does it say, it says that the system will respond fast enough. How fast, in less than 1 second, and not always not very little, but most of the time. The "most" is perfectly quantified it says 90 percent of the time. So, this is a typical example requirement of performance.

(Refer Slide Time: 25:28)



Performance testing techniques

- Load testing: It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are monitored.
- Stress testing: It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.

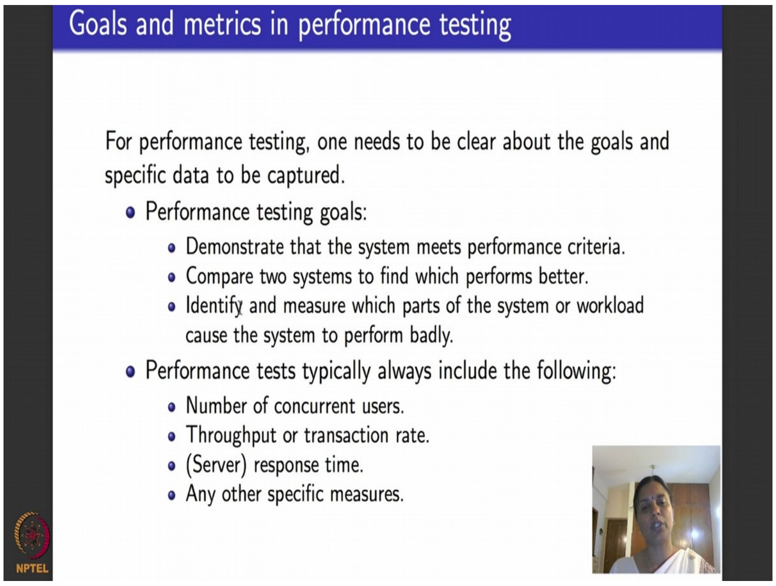
So, when a performance test for this requirement, I create data that check helps me to validate whether this requirement is met or not. So, typical performance testing techniques are classified as follows. These are the most popular ones, but again, they are not exhaustive, there are some more that I have not listed in these slides. Load testing is a performance testing technique. What does it do? It is in fact the simplest performance testing technique that is conducted to understand how a system behaves under a certain load. Like I told you, right, if I have IRCTC system a peak load, let us say for such a system occurs during Tatkal booking hours, load testing checks if the system actually caters to that peak load. It will result in measuring important business critical transactions, load on the database applications server and so on.

Stress testing is another kind of performance testing. It is performed to find the upper limit capacity of the system and also to determine how the system will perform if it is average current load goes well above the expected maximum. Let us say, there is a particular system. Let us say a system that displays the marks of the results of a higher secondary school examination. You can imagine that the system will be very stressed for the first few hours, after the results are out, right. So, what they do is that let us say it is supposed to cater to a 100000 students accessing their marks soon after the results are out, when this stress test it, they will test it for a limit above 100000 to see how the system actually responds to it.

They will stress the system in terms of giving inputs to the system and see if the system is able to cater to it or if it fails, how exactly does it fail. The next kind of performance testing is what is called soak testing also known as endurance testing. This is performed to determine system parameters under continuous expected load. So, you understand the meaning of the word soak, right. Soak means what I keep it soaked, I see how, if there is a steady load that comes in continuously of a standard number of users periodically without a break, does the system perform as it is expected to?

So, that is what soak test does, typically, it is useful for how much memory is utilized and whether there are memory leaks or not. Then there is something called spike testing which is a kind of stress testing. So spike testing is like stress testing, but what it does is that it increases the number of users suddenly. There is a sudden spike, not a gradual spike, and see how well the system performs under the sudden increase in the number of users. The increase in the number of users need not be above the maximum number, but it is a fairly sudden increase. So, the main aim is to determine whether the system will be able to retain the sudden increase in the workload.

(Refer Slide Time: 28:29)



Goals and metrics in performance testing

For performance testing, one needs to be clear about the goals and specific data to be captured.

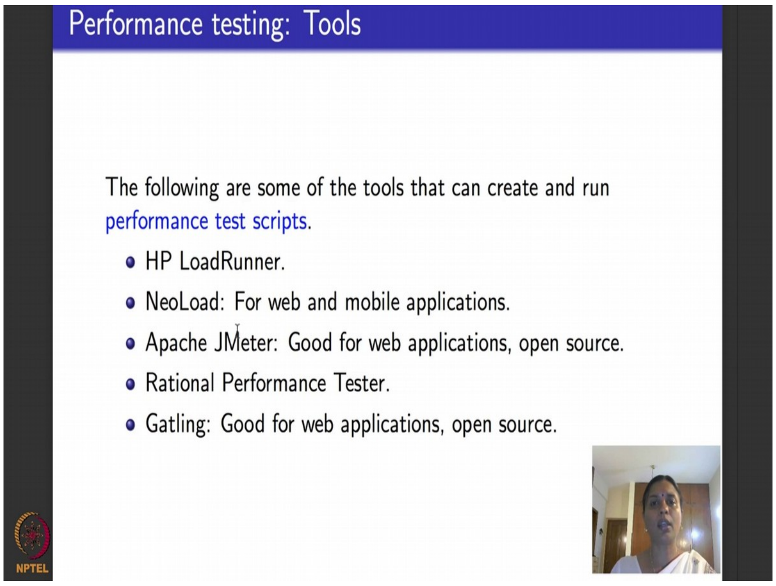
- Performance testing goals:
  - Demonstrate that the system meets performance criteria.
  - Compare two systems to find which performs better.
  - Identify and measure which parts of the system or workload cause the system to perform badly.
- Performance tests typically always include the following:
  - Number of concurrent users.
  - Throughput or transaction rate.
  - (Server) response time.
  - Any other specific measures.

So, I have covered only 4, but there are several other performance tests as I told you which I have not really not mentioned or covered here. So, for performance testing, typically, it is important to know what is it that you are going to measure, what are your goals, what is it the data that I want to capture, what am I going to measure because if you go all over the place there are so many parameters you can capture, so much data you can capture, typically as a tester, you will be left with an overdose of information which too much of data that cannot be analyzed.

So, it is important to find out what is the kind of performance testing that one is going to do, currently doing and for this kind of performance testing what is the precise goals I have, what is the set of data that I want to measure to be able to achieve this goals. So, typically, there are 3 kinds of performance testing goals that people can have. You could take a single system and demonstrate that the system meets its specified performance criteria. You could compare 2 different systems of the same kind and determine which performs better than the other with reference to a specified set of criteria or you could identify and measure the parts of a system or workload that causes a system to perform badly.

The violating parameters you could focus on that. So, typically whatever you do, performance tests will include one of these or most of these parameters. Let us say most of the systems are meant to interact with users. So, it will say how many number of concurrent or parallel users are using a system, that is one of the things. Then what is the throughput or transaction rate that the system is committing to and what is the server response time. In addition to these three, any other specific measures that you want to include can be included.

(Refer Slide Time: 30:13)



Performance testing: Tools

The following are some of the tools that can create and run performance test scripts.

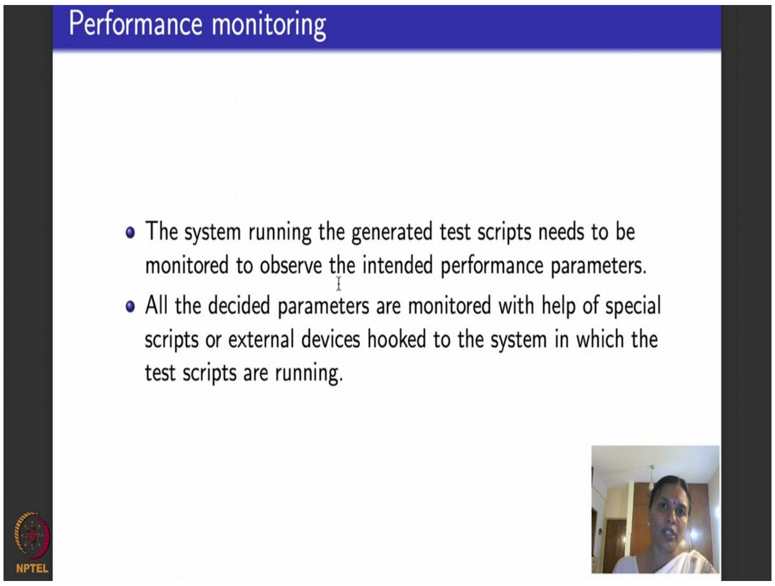
- HP LoadRunner.
- NeoLoad: For web and mobile applications.
- Apache JMeter: Good for web applications, open source.
- Rational Performance Tester.
- Gatling: Good for web applications, open source.

So, here are some performance testing tools that are very popular? some of them are proprietary, some of them are open source, I have marked those that are open source against the tools, the rest of them are proprietary. The most popular performance testing tool in my opinion is that by HP, it is called load runner, that is what is best. Before I move on, what do these kinds of tests do?

They basically generate scripts for execution. They generate large number of performance scripts based on your target of performance measurement and they also execute these scripts that is the purpose of these tools. The popular one is by HP load runner been there for many years. There is one more tool called Neo load which is useful for web and mobile applications performance testing. There is a tool called JMeter which is good for web applications, this is an open source tool. There is another tool called rational performance tester again a proprietary tool there is a tool called Gatling which is again good for web applications and open source.

If you recap of my lectures on web applications, I had given you a link to another a web applications online testing tool. That tool is an exhaustive, that page is an exhaustive listing and includes several other performance testing tools also. Here I have just put some of the ones in particular HP LoadRunner and JMeter, I have tried out. The rest I have heard others using it. So, I thought I will put this list. So, typically after these performance testing tools generate and execute the scripts, you might have decided on the parameters that you want to monitor like the ones that I mentioned here.

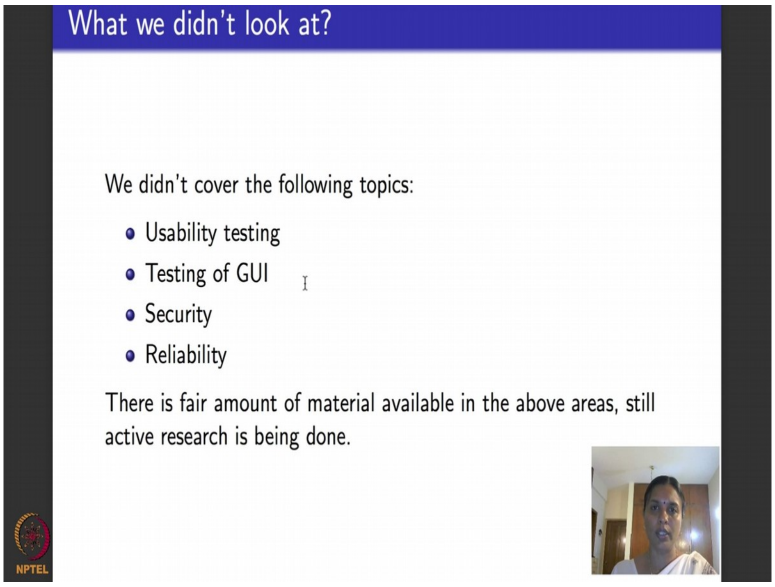
(Refer Slide Time: 31:55)


Performance monitoring

- The system running the generated test scripts needs to be monitored to observe the intended performance parameters.
- All the decided parameters are monitored with help of special scripts or external devices hooked to the system in which the test scripts are running.

So, you; how do you monitor these parameters? You actually hook the system to external things, you are a separately run off loaded scripts or drivers that will monitor the system or even hardware devices that are plugged into monitor these parameters. They are monitored as I told you with the help of special, dedicated scripts that do not run as a part of the main system, but run separately so that the performance of the main system is not affected. Or they could even be monitored with the help of external devices that are hooked to the system in which the test scripts are running. So, that is the brief overview of performance testing and non-functional testing. I will go back to the slide that I began with the taxonomy slide. We did cover an overview of all of these I will do one separate lecture on regression testing. It also helps to know what I did not do before we end this module on non functional system testing

(Refer Slide Time: 32:46)



What we didn't look at?

We didn't cover the following topics:

- Usability testing
- Testing of GUI
- Security
- Reliability

There is fair amount of material available in the above areas, still active research is being done.

So, we did not look at usability testing at all, very important very significant area. There is a separate testing for graphical user interface. Again extensive algorithms based testing is available based on finite state machines and so on. We have not looked at any of those and as I have told you 2 or 3 times, by now we have not at all looked at security and reliability testing in detail. So, it is to be noted that there is a fair amount of material available in all these areas and there is active research still going on, but we have not been able to cover them in the course. So, I will end here and I will come back with a module on regression testing.

Thank you.