

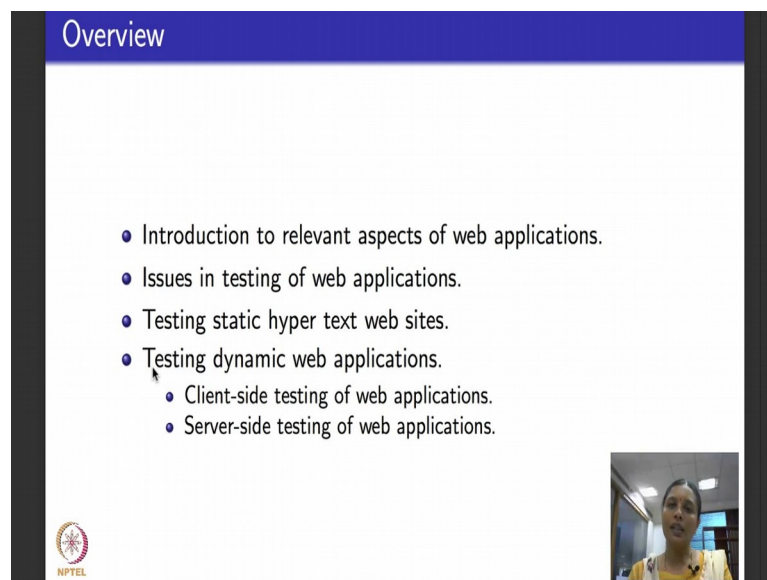
Software testing
Prof. Meenakshi D'Souza
Department of Computer Science and Engineering
International Institute of Information Technology, Bangalore

Lecture – 45
Testing of Web Applications and Web Services

Hello again, welcome to week 10. We are going to start on something completely new this week. As I told you towards the end of the last lecture of week 9 we saw what we did till now we mainly did coverage criteria and so various kinds of testing using this model or that structure--- graphs, logic, sets and so on. This week I am going to take classes of applications we are going to start with web applications web software move on to object oriented applications, we learn a bit about those applications, we learn problems that are unique related to testing about each of these applications. We will compare and see the coverage criteria that we have learnt so far whether they can be directly used to test these applications or not and if not I will tell you new algorithms that can be specifically used to test applications.

So, we are going to begin this week by looking at testing of web applications.

(Refer Slide Time: 01:05)



Overview

- Introduction to relevant aspects of web applications.
- Issues in testing of web applications.
- Testing static hyper text web sites.
- Testing dynamic web applications.
 - Client-side testing of web applications.
 - Server-side testing of web applications.

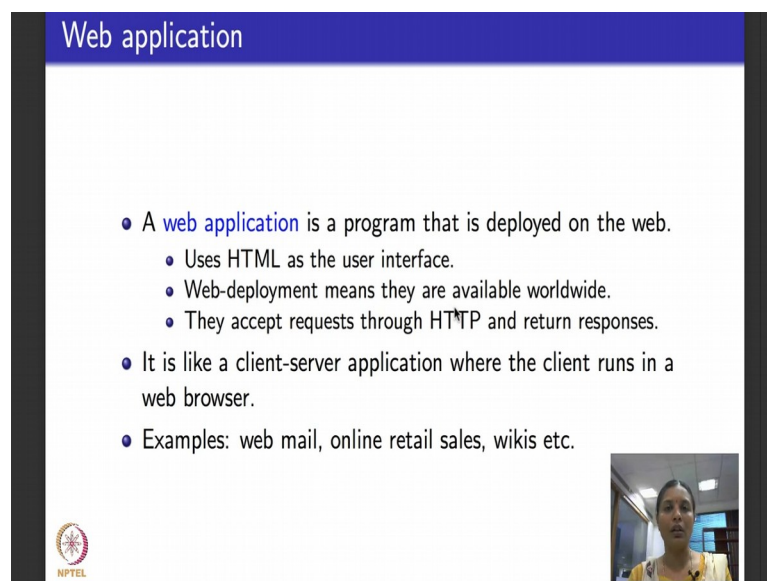
NPTEL

So, this is an overview of what we are going to do over the next few lectures. Today I will introduce you to relevant aspects of web applications we will see what they are and

then we will also discuss what are the specific issues that come with testing of web applications.

And then when it comes to testing it helps to have the following broad categorization we will test what are called static hypertext websites, websites that remain the same, look the same to almost every user who accesses it all the time. Basically have nothing more than HTML files. And then when it comes to testing its important to separately consider dynamic web applications. In dynamic web applications the content of the web page is created dynamically by an underlying application or a program. Here usually the kind of code that is written for the server side is very different from the kind of code that is done for the client side. Hence the testing that we have to do for the client side and server side are also very different. So, this is will, this is a structure of what we are going to see about testing of web applications put together.

(Refer Slide Time: 02:12)



The slide is titled "Web application" in a blue header. It contains a list of bullet points defining web applications. In the bottom left corner, there is an NPTEL logo. In the bottom right corner, there is a small video inset showing a woman speaking.

- A **web application** is a program that is deployed on the web.
 - Uses HTML as the user interface.
 - Web-deployment means they are available worldwide.
 - They accept requests through HTTP and return responses.
- It is like a client-server application where the client runs in a web browser.
- Examples: web mail, online retail sales, wikis etc.

Today, I will begin with introducing you to the relevant aspects of web applications. Please remember that this is not meant to be a comprehensive introduction to what a web application or a web software is. Feel free to look up other courses or online material for that. I will just give you a bird's eye overview of what it is and consider it as understood from the point of view of testing. Our issue will be to focus on testing not to thoroughly introduce you to web applications.

So, what is a web application for us? For us a web application is a program written in some language right, but the program, difference is not deployed in the computer it is deployed on the web. What do we mean by its deployed on the web? It is basically deployed in a computer that is meant to act as a server for further deployment or availability on the web or the internet. What is the user interface that this kind of web application uses its basically HTML stands for hypertext markup language, standard language that people use to write web pages. Web deployment means what, its available on the internet worldwide as long as you have access to it, you have access to this website.

And typically a web application, how does it talk to a user, an end user? It talks to an end user through a server that accepts requests from an end user or a client through HTTP - HTTP stands for hypertext transfer protocol. It is a standard protocol that is used in the internet and it also returns responses which would websites basically your information provided by a web page through HTTP again. So, this is a web application for us. So, it is a lot like a classical client server application, the only difference is the server is a web server meant to host the application on the internet and the client runs in a web browser. What a web browsers? Internet explorer, Mozilla, Google Chrome and so on.

So, here are examples of classical web applications. We all use web mail some form of the other--- Gmail, Microsoft office outlook, yahoo mail, hotmail and so on. We also use a lot of online retail sales--- amazon, flipkart and so on and we use wikis these are various categories of web applications and lots and lots more are also there.

(Refer Slide Time: 04:28)




The slide has a blue header with the text "Web services". Below the header, there is a bulleted list defining a web service. In the bottom right corner, there is a small video inset showing a person. The NPTEL logo is in the bottom left corner.

- A **web service** is a web-deployed program that accepts XML messages wrapped in SOAP.
 - Usually no user interface with humans.
 - Service must be published so other services and applications can discover them.

There are a slightly special class of web applications called web services. What is a web service? Web service like a web application is another program that is deployed on the web, it accepts XML messages that are wrapped in SOAP or restful services. Usually it has no interface with human it directly talks to programs or clients and the service must be published so that other services and applications can discover a web service.


In our lectures when we focus on testing we will not really explicitly consider web services as a standalone separate kind of web application, we look at generically testing web applications as we introduce them here.

(Refer Slide Time: 05:10)



General characteristics of web software

- Composed of independent, loosely coupled software components.
 - All communication is through messages.
 - Web application messages always go through clients.
 - The only shared memory is through the session object - which is very restricted.
 - The definition of state is quite different.
- Inherently concurrent and often distributed.
- Most components are relatively small.
- Uses numerous new technologies, often mixed together.
 - JSP, ASP, Java, JavaBeans, Javascript, Ajax, PHP etc.



So, what are the general characteristics of such web applications or web software? I will use the term web app, web application, web software all these terms synonymously, they all mean the same for us. Web application is composed of several independent loosely coupled program components.

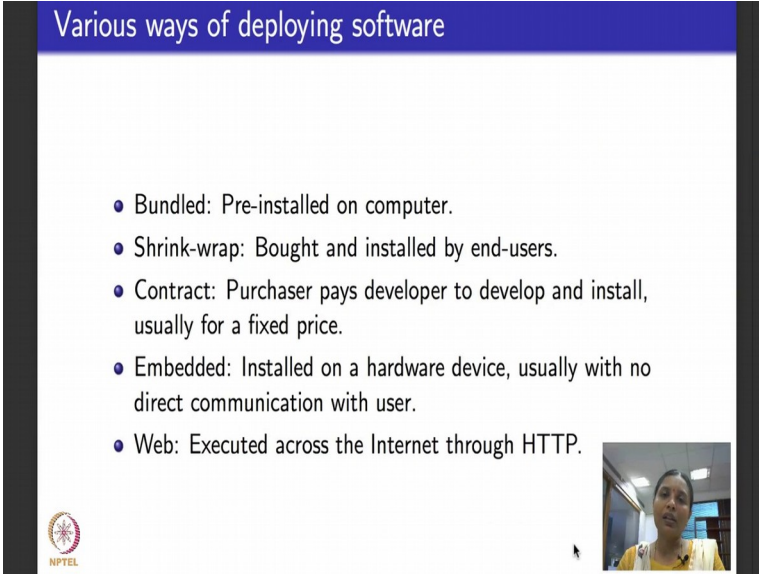
How do these loosely coupled program components talk to each other? How do they constitute one web application? They communicate with each other through message passing, the interface that they use is message passing interface and these messages from the application always go through dedicated entities called clients and sometimes they do share memory, but it is very restricted it is through what is called session objects and the definition of a state of a program we know what it is. It means the class of all set of all variables of a program and a location counter, but here the state of a program as a web application is usually quite different, does not correspond to the classical notion of a program state.

As I told you because it consists of independent loosely coupled components a web application is a distributed or a concurrent application and it has several components each of them very small and not only that each of these components could be developed or written processed using a completely different technology. For example, somebody could do JSP Java server pages somebody could do ASP, somebody could just write

Java, Javabeans, Javascript, Ajax, PHP, HTML. If you go and search for web technologies and web applications you will find all these terms through and across.

My goal again, to reiterate, is to not to exhaustively introduce you to these terms, but to more bring it up as one of the characteristics of web applications. That when we put them all together to test we have to handle in our tests several different technologies that come from completely different domains, that is important to remember.

(Refer Slide Time: 07:09)



The slide is titled "Various ways of deploying software" in a blue header. It contains a bulleted list of five deployment methods. In the bottom left corner is the NPTEL logo, and in the bottom right corner is a small video inset showing a person speaking.

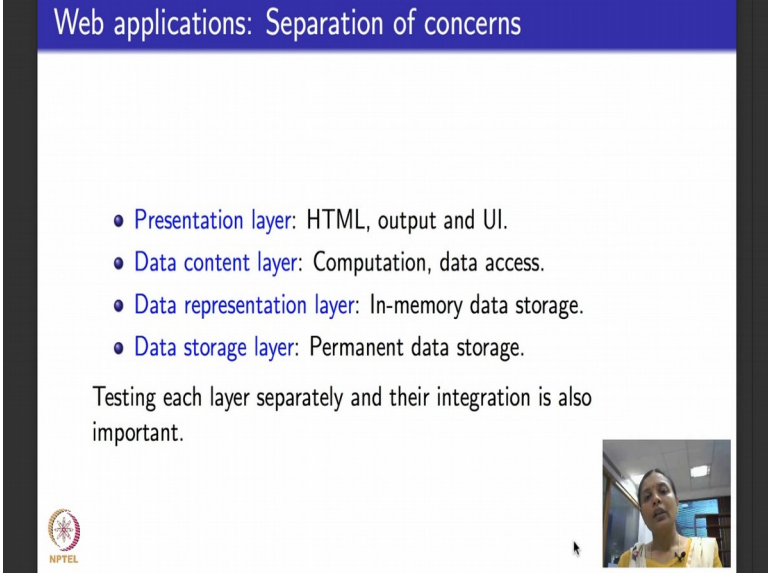
- Bundled: Pre-installed on computer.
- Shrink-wrap: Bought and installed by end-users.
- Contract: Purchaser pays developer to develop and install, usually for a fixed price.
- Embedded: Installed on a hardware device, usually with no direct communication with user.
- Web: Executed across the Internet through HTTP.

Typically when I consider a software any piece of software that you have what are the various ways of deploying it? This is not specific to web applications or web software, any generic software you could bundle a software in which case its usually pre installed on a computer.

Like for example, if you buy a package, if you buy a computer then the operating systems in the computer comes bundled along with it its pre installed the way you want it. You could shrink wrap a typical example for this is you buy the software and install it like an anti-virus software that you would buy and install. Or you would have a contract software where you have access to the software, but for a specified period of time that you buy the contract for and you could have embedded software which is usually installed on a specific purpose special purpose hardware device like in a microwave oven in a car, in an aeroplane and so on. And it typically directly communicates only with sensors and actuators and the embedded device, does not communicate with the user.

And then now the focus of this lecture is what is called web software. How is it deployed? It is executed across the internet through HTTP protocol typically hosted by a dedicated server for this purpose called the web server.

(Refer Slide Time: 08:22)



Web applications: Separation of concerns

- **Presentation layer:** HTML, output and UI.
- **Data content layer:** Computation, data access.
- **Data representation layer:** In-memory data storage.
- **Data storage layer:** Permanent data storage.

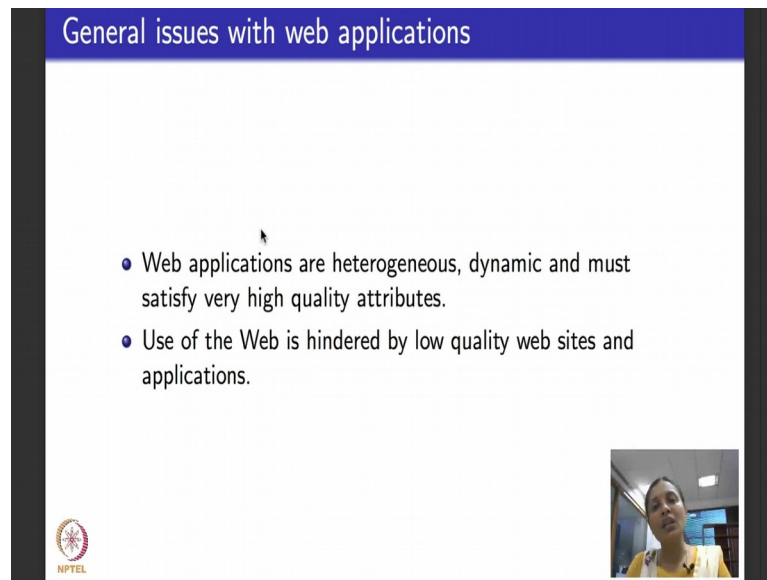
Testing each layer separately and their integration is also important.

NPTEL

Now, let us look at web application, consider the typical architecture of a web application. The architecture has several layers where separation of concerns happen. There is something called the higher, one of the higher layers of the architecture is a presentation layer where the HTML and UI output are visible. Then comes the layer that contains the data called the data content layer where the actual computation and access to data happen.

Below the data content layer is a data representation layer where the data is stored and retrieved in memory and finally, you have data storage layer which represents permanent data storage that happens because of a web application. Testing each layer separately is important and what is also important is to do system level testing, where you test for integration across these layers that do separation of concerns in a web application. Here are some generic issues with web application? They, as I told you, are heterogeneous dynamic distributed and because they are available freely across the internet typical expectation is that they must be of very high quality. They cannot be slow, they cannot have a bug, they have to be reliable, they need to respond very quickly.

(Refer Slide Time: 09:16)



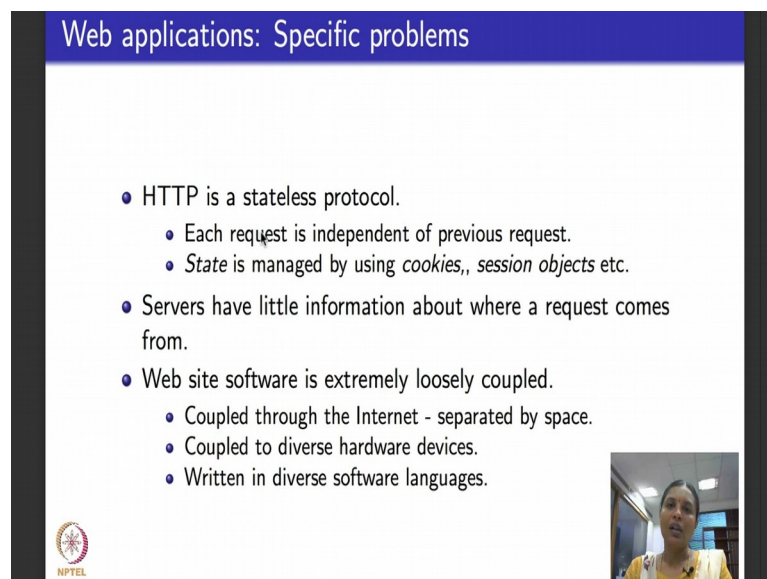
General issues with web applications

- Web applications are heterogeneous, dynamic and must satisfy very high quality attributes.
- Use of the Web is hindered by low quality web sites and applications.

The slide features a blue header with the title 'General issues with web applications'. Below the title, there are two bullet points. In the bottom right corner, there is a small video inset showing a woman speaking. The NPTEL logo is visible in the bottom left corner.

So, their quality attributes and expectations from the software is very high, but you while know by experience that there are so many low level software that actually hinder the use of these web applications.

(Refer Slide Time: 09:49)



Web applications: Specific problems

- HTTP is a stateless protocol.
 - Each request is independent of previous request.
 - State is managed by using *cookies*, *session objects* etc.
- Servers have little information about where a request comes from.
- Web site software is extremely loosely coupled.
 - Coupled through the Internet - separated by space.
 - Coupled to diverse hardware devices.
 - Written in diverse software languages.

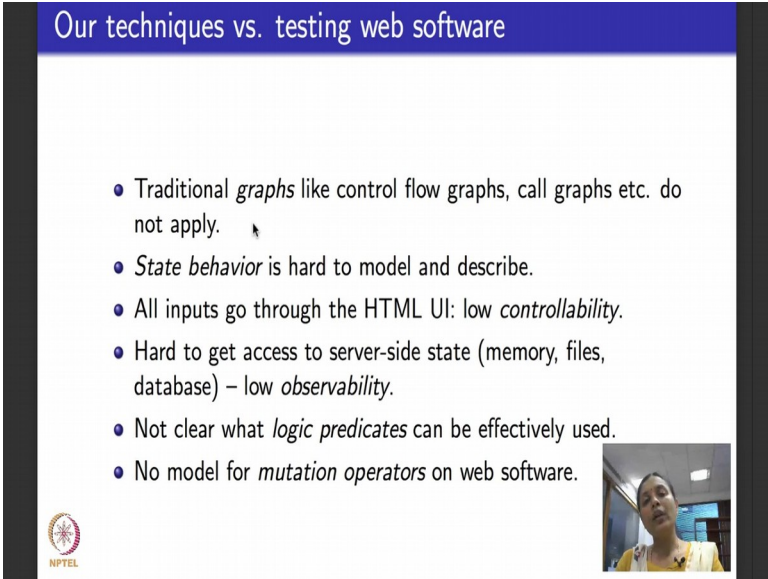
The slide features a blue header with the title 'Web applications: Specific problems'. Below the title, there is a list of bullet points. In the bottom right corner, there is a small video inset showing a woman speaking. The NPTEL logo is visible in the bottom left corner.

Now, let us move on and discuss specific issues related to web applications. As I told you all web applications and web software use hypertext transfer protocol HTTP. What do we know about HTTP, it is a stateless protocol.

What do we mean by it is a stateless protocol? Suppose there is a new HTTP request it is completely independent of the history. It is completely independent of the previous requests, the requests before that and so on. So, if it is a stateless protocol how do people manage state when they deploy web application? You all might have heard of the term cookies, session objects and so on right. So, those are basically entities that web programmers or web app developers use to manage state information whenever needed.

The next is, web server typically has very little information about where a request comes from. It could be from malicious client, it could be from a genuine client. Typically you do lot of authentication and authorization, but still there is very little information about where exactly the request is coming from and what happens to the response in turn. Web software as we discussed earlier its extremely loosely coupled right, extremely loosely, there is importance to that word why, there is something called tightly coupled and loosely coupled here I have one more qualifying adjective, I say it is extremely loosely coupled. Why is that so, because it is coupled through the internet which means what the software is physically separated in space. It does not reside in one entity, it does not reside within one server. It is also coupled to diverse hardware devices, it could be anywhere in any part of the world and as I told you its written using diverse technologies and a diverse set of software languages.

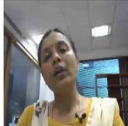
(Refer Slide Time: 11:36)



Our techniques vs. testing web software

- Traditional *graphs* like control flow graphs, call graphs etc. do not apply.
- *State behavior* is hard to model and describe.
- All inputs go through the HTML UI: low *controllability*.
- Hard to get access to server-side state (memory, files, database) – low *observability*.
- Not clear what *logic predicates* can be effectively used.
- No model for *mutation operators* on web software.

NPTEL



Let us look at all the techniques that we have learnt so far and analyze one at a time about whether any of them will be useful for testing web software. What did we begin the course with? We taught graphs to start with, but what were the graph models of software artifacts that we considered? We considered control flow graph, we considered call graphs and so on. If you see none of these kind of models apply for system level testing of web application.

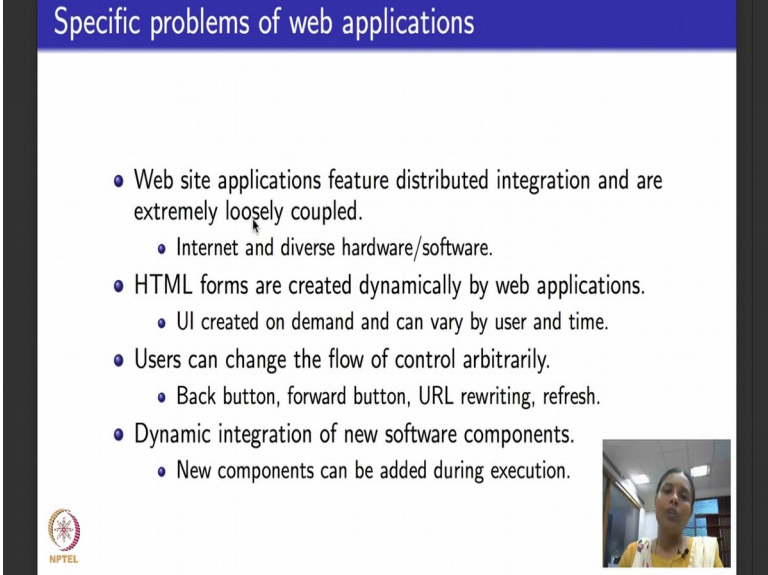
Of course you could consider one individual program that is written as a web application and if you are doing unit testing for that program then you could consider the control flow graph do your coverage criteria. But that is not what we talking about, we talking about system level testing of a web app. In that case working with control graph or call graph does not really apply because the program is distributed across the internet. The other thing is, we also learnt about state machines, the specific graph models that model the state behavior of software. Here state behavior of a web application, if you think about it, there is some part of the server state, there is some part of the client state, different clients have different states. It is quite a hard task to model the state.

Typically all, the next important issues are related to RIPR model or controllability and observability. Typically both controllability and observability for web software is very low. Why is controllability low, because all inputs please remember go through HTML files right. So, if I have to control an input I am talking about controlling the entire HTML file of an entire webpage. It is going to be difficult to provide inputs that can be controlled within a user, and once you provide an input, let us say you manage to provide an input, you managed to get controllability, the next is to observe what happens in this for the state of a web program which is hosted in a web server. It is very difficult to get access to server side details. For good reasons it is not secure enough and typically do not want to, we do not know where the memories are, where the file is where the database is. So, from the point of view of testing observability and controllability becomes very low, so testing becomes difficult.

Now the other kind of software artifact that we learnt were logic predicates, but it is not clear what kind of logic predicates it can be effectively used to do system level testing. Again the last one that we learnt was related to mutation operators. To the extent that I am aware of there are no known mutation operator specifically for web software. So, it looks like most of the techniques that we learnt cannot be directly put to use. We will use

graph based testing to some extent, but only for restricted websites that are static in nature. The rest of them we really have to understand or develop pretty new techniques that could be based on the techniques that we learnt so far, but none of the ones that we have learnt so far directly apply.

(Refer Slide Time: 14:35)



The slide is titled "Specific problems of web applications" in a blue header. It contains a bulleted list of five points, each with a sub-bullet. In the bottom left corner is the NPTEL logo, and in the bottom right corner is a small video inset showing a person speaking.

- Web site applications feature distributed integration and are extremely loosely coupled.
 - Internet and diverse hardware/software.
- HTML forms are created dynamically by web applications.
 - UI created on demand and can vary by user and time.
- Users can change the flow of control arbitrarily.
 - Back button, forward button, URL rewriting, refresh.
- Dynamic integration of new software components.
 - New components can be added during execution.

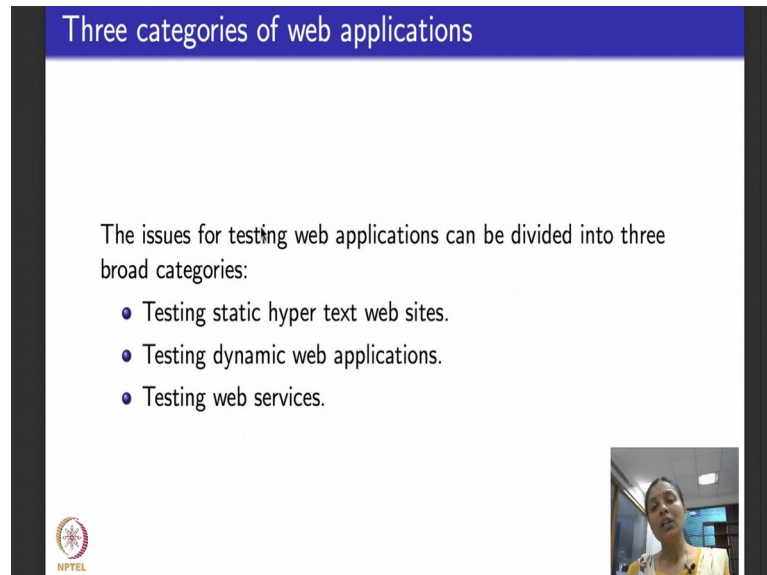
So, now we will revisit the specific problems of web applications because it is useful to know when we are designing testing techniques. So, as I told you they are very extremely loosely coupled, over the internet, this diverse hardware and software and I told you HTML forms are created dynamically by web applications.

So, there is very little controllability and observability on the input. The user interface that is provided through a HTML file can vary and it is created on demand based on the credentials, and users can change the flow of control arbitrarily in the middle of a transaction as a page is loading somebody could press a back button or a forward button could press a refresh button. That is why when you handle secure transactions like bank and that you explicitly get messages saying do not press back button. But typically for a normal web application there is no expectation that nobody will press the back buttons.

So, as your testing while your input is executing, while your program is running if the control changes then what you observe may not actually correspond to the actual web app or the web software behavior at all. So, it is quite a dicey thing. And the other thing

is that dynamically new software components can be added into a web application. So, all these are issues.

(Refer Slide Time: 15:53)



The slide is titled "Three categories of web applications" in a blue header. The main content area is white and contains the text: "The issues for testing web applications can be divided into three broad categories:". Below this text is a bulleted list with three items: "• Testing static hyper text web sites.", "• Testing dynamic web applications.", and "• Testing web services.". In the bottom left corner, there is a small NPTEL logo. In the bottom right corner, there is a small video inset showing a woman speaking.

Three categories of web applications

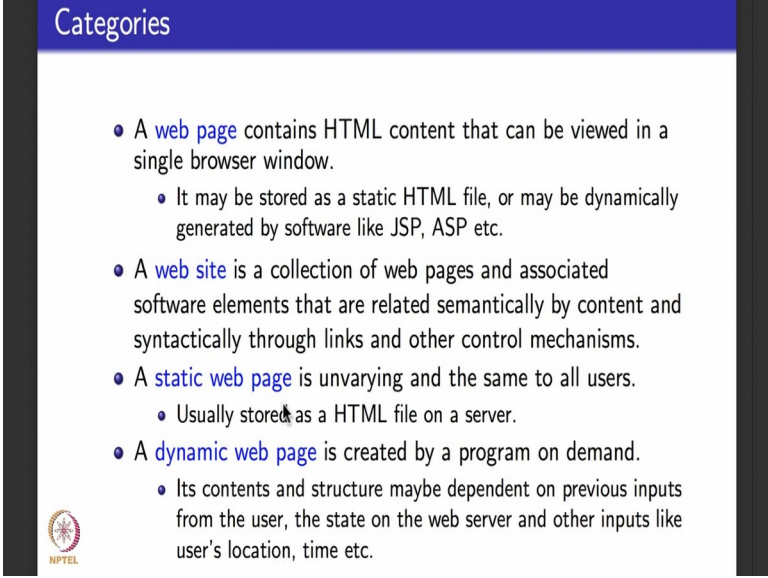
The issues for testing web applications can be divided into three broad categories:

- Testing static hyper text web sites.
- Testing dynamic web applications.
- Testing web services.

NPTEL


When it comes to testing we prefer to bucket web applications into three different categories. We will separately deal with what we call static hypertext websites which are basically plain HTML files that are written once, put on the web server deployed and not touched. And then will separately deal with testing dynamic web applications. On that will do separately client side testing and separately server side testing. As I told you we really will not exclusively consider testing of web services. We will generically deal with testing of web applications. The last topic here we will not specifically consider it.

(Refer Slide Time: 16:33)



Categories

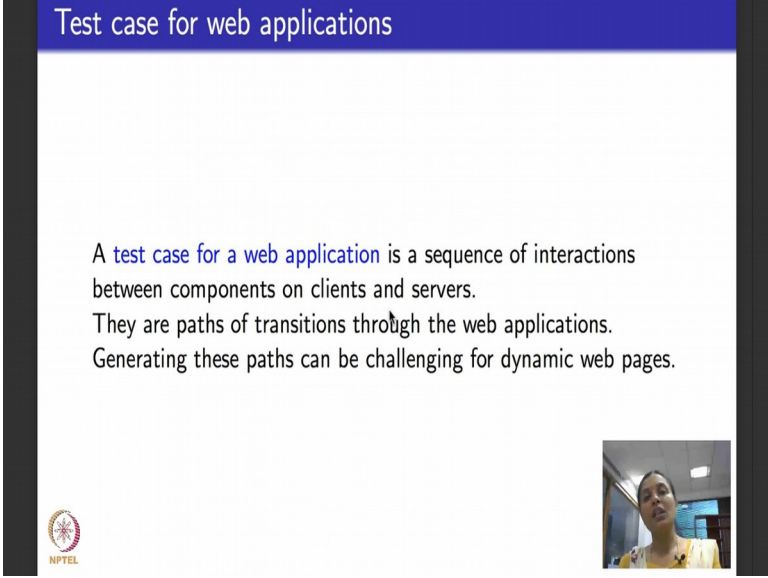
- A **web page** contains HTML content that can be viewed in a single browser window.
 - It may be stored as a static HTML file, or may be dynamically generated by software like JSP, ASP etc.
- A **web site** is a collection of web pages and associated software elements that are related semantically by content and syntactically through links and other control mechanisms.
- A **static web page** is unvarying and the same to all users.
 - Usually stored as a HTML file on a server.
- A **dynamic web page** is created by a program on demand.
 - Its contents and structure maybe dependent on previous inputs from the user, the state on the web server and other inputs like user's location, time etc.



So, just to re-define find a few terminologies that we will need for the rest of the few lectures. For us a web page contains HTML content and it is typically viewed in a browser single browser window. It can be a static HTML file or it could be dynamically generated by using JSP, ASP or any other kind of web software.

A web site is a collection of web pages, static and or dynamic and its associated software element that are related semantically in a meaningful way by their content. How are they related syntactically? They are related syntactically by hyperlinks and other mechanisms for passing control from one web page to the other--- hyperlinks forms and so on. A static web page never changes, it is the same to all the users all the time as long as it is available, usually stored as a HTML file on a dedicated web server. In contrast, a dynamic web page is created by a program or a web program web software on demand, it contents and structure typically depend on previous inputs from the user, state of the web server, other inputs like where the user is from, what time of the day, what is the data provided by the user and so on.

(Refer Slide Time: 17:53)



The slide has a blue header with the text "Test case for web applications". The main content area is white and contains the following text:

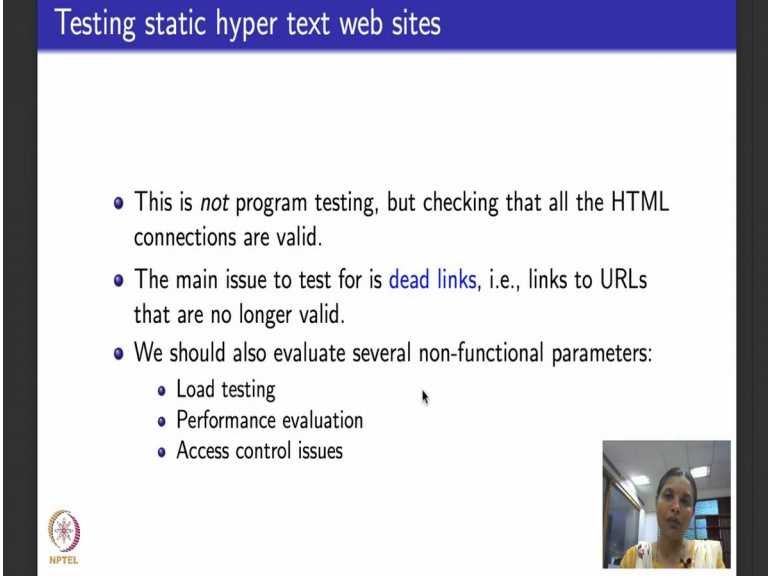
A test case for a web application is a sequence of interactions between components on clients and servers.
They are paths of transitions through the web applications.
Generating these paths can be challenging for dynamic web pages.

In the bottom right corner, there is a small video inset showing a person speaking. In the bottom left corner, there is a small circular logo with the text "NPTEL" below it.

For us when we test web applications, it helps to first think back and understand what will a test case for a web application be. A test case for a graph will be a path, a test case for a logical predicate will be true or false values to the inputs. A test case for mutation based testing would be any value that kills the mutant and so on. Similarly what is a test case for a web application? A test case for a web application is basically a sequence of interactions between components that reside on the clients and the components that reside on the server. How are they given? Test case is usually depicted by a path that contains states and transitions through the various web applications. These paths have to be generated for static and dynamic websites. For static web pages, it is easy to do. For dynamic websites it can be challenging mainly because as I told you, you could have users changing the flow of control arbitrarily. So, if you generate a particular dynamic web content, halfway through the content being fed into the software as an input, the nature of the software itself could be changed by the user.

So, just as a recap what is the test case a test case is basically a sequence of web pages which are given as transitions through the web applications; easier to create for static websites little more difficult to create for dynamic websites.

(Refer Slide Time: 19:24)



The slide has a blue header with the title "Testing static hyper text web sites". The main content area is white and contains a bulleted list. The first bullet point states that this is not program testing but checking for valid HTML connections. The second bullet point discusses testing for dead links. The third bullet point lists non-functional parameters: load testing, performance evaluation, and access control issues. In the bottom left corner is the NPTEL logo, and in the bottom right corner is a small video inset of a woman speaking.

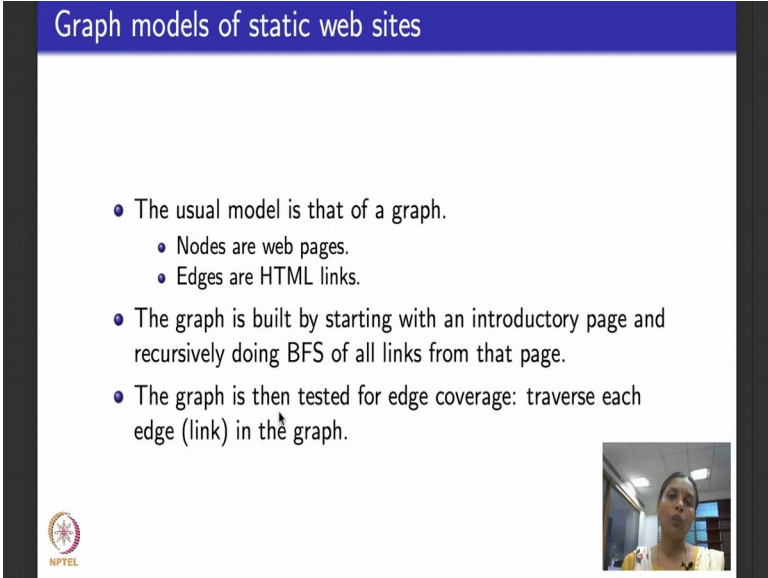
- This is *not* program testing, but checking that all the HTML connections are valid.
- The main issue to test for is **dead links**, i.e., links to URLs that are no longer valid.
- We should also evaluate several non-functional parameters:
 - Load testing
 - Performance evaluation
 - Access control issues

So, we will begin by looking static websites in this lecture. Next lecture I will begin by doing dynamic websites. Since static website what is a web page look like there is no program involved at all please remember that, maybe there is some scripting involved the ultimate contents to test is a set of plain HTML files that never fall. This is not program testing because there is no program. But what is this testing do? It basically checks that all the HTML connections are valid right. If one hyperlinks another one that connection is valid and it goes to the intended page, not only does it go to the intended page, you also check that the particular hyperlink or a connection that is there available from a website does not go to any URL that is dead or defunct or that are no longer valid.

So, such kind of testing should also include testing for what are called dead links. Dead links are those that are links, hyperlinks to URLs that are not valid. Typically when it comes to system level testing of web applications people also evaluate several non functional parameters. They test the load of a system, classical example would be let us say use IRCTC web application, load testing would be to let us exercise maximum number of users that the system can scale up to. Let everybody log in and see if the system is still able to smoothly let the person transact and book his or her ticket the way he or she wants. Another thing is performance evaluation which basically talks about the fast, the efficiency of response to of a web application.

The third is access control issues in the sense that you need certain kind of accesses to be able to access certain kind of dynamic web page. Like for example, if you are accessing your bank webpage before you actually get down to the dynamically created content that gives you your user account details, your access as a user needs to be authenticated by means of a login and a password. So, system level testing of web applications also include these non functional parameters. Our focus in these lectures would be on functional testing of system level parameters. So we will not really look at load or performance testing in this course.

(Refer Slide Time: 21:41)



The slide is titled "Graph models of static web sites" in a blue header. It contains three bullet points:

- The usual model is that of a graph.
 - Nodes are web pages.
 - Edges are HTML links.
- The graph is built by starting with an introductory page and recursively doing BFS of all links from that page.
- The graph is then tested for edge coverage: traverse each edge (link) in the graph.

In the bottom left corner is the NPTEL logo. In the bottom right corner is a small video inset showing a woman speaking.

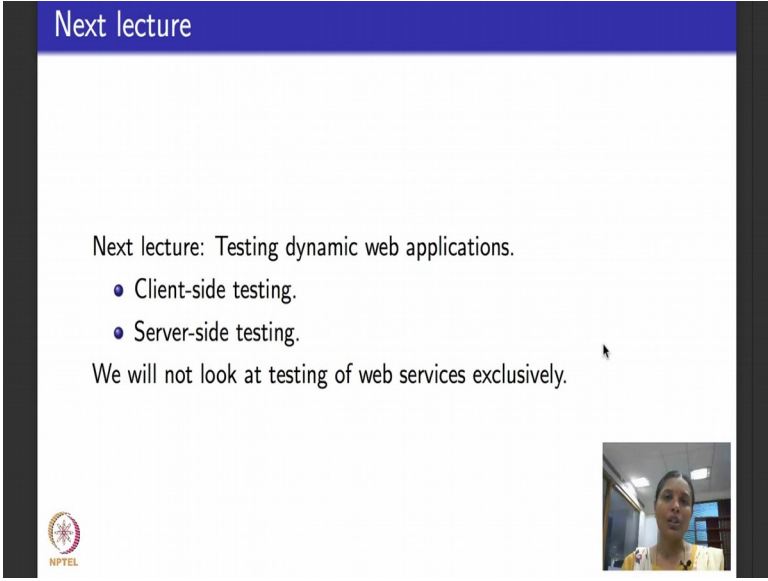
So, when it comes to static website what is the graph, what is the model of testing that we use? It is not too difficult to see that graph models as we saw will always work. Of course, this is not call graph or a control flow graph, this is a different kind of graph. What will be the nodes or vertices of this graph? The nodes or vertices of these graphs are the various web pages, each individual page. What are the edges of these graphs? Let us say you are in one particular page, there is a hyperlink, you click on that hyperlink you go to another page. Then you say that there is an edge from the node corresponding to the first webpage to the node corresponding to the second web page. So, edges are HTML links.

And how do you build such a graph? You typically build the graph by starting from an introductory page. In static pages they will usually be a index dot HTML or a welcome

dot HTML which is the beginning page of the static website. You start from there do a breadth first search, recursively, by looking at each link from that page and then go to the pages that it links to, again do a BFS to the links of that page and so on till there are no further links to be explored. Once this graph is generated what is the main aspect that is tested? The main aspect that is tested is classical edge coverage. What do edges of this graph represent? Is each link working fine?

So, edge coverage basically tests for traverse each hyperlink in the static web site. So, testing static web sites, that is about it there is nothing much. You generate a graph vertices of the graph are web pages, edges of the graphs are hyperlinks and then you do edge coverage in the graph. This basically tests for static website working correctly, because there is nothing like performance load, there is no dynamic data. That is it we are done with system level testing of static websites.

(Refer Slide Time: 23:29)

A presentation slide titled "Next lecture" in a blue header. The main content area is white and contains the text "Next lecture: Testing dynamic web applications." followed by a bulleted list with two items: "Client-side testing." and "Server-side testing." Below the list, it says "We will not look at testing of web services exclusively." In the bottom left corner is the NPTEL logo, and in the bottom right corner is a small video inset showing a woman speaking.

Next lecture

Next lecture: Testing dynamic web applications.

- Client-side testing.
- Server-side testing.

We will not look at testing of web services exclusively.

NPTEL

Now, the next lecture what I will tell you is let us move on and look at dynamic web applications. When it comes to dynamic web applications, as far as testing is concerned as I told you it helps to distinguish between the client side testing and server side testing. So, we look at these two independently and see some new techniques or technologies or algorithms that have been developed for client side testing and for server side testing and as I told you a couple of times before in this lecture, we will not look at testing web services exclusively.

So, I will come back to you in the next lecture for testing a dynamic websites.

Thank you.