

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

**Lecture - 47**  
**Testing of Web Applications and Web Services**

Hello we are in the middle of week 10. I was doing web applications testing.

(Refer Slide Time: 00:17)



The slide is titled "Overview" and contains a bulleted list of topics. The list includes: Introduction to relevant aspects of web applications, Issues in testing of web applications, Testing static hyper text web sites, and Testing dynamic web applications. The last item, "Testing dynamic web applications," has two sub-bullets: "Client-side testing of web applications" and "Server-side testing of web applications." The slide also features the NPTEL logo in the bottom left corner and a small video feed of the professor in the bottom right corner.

- 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.

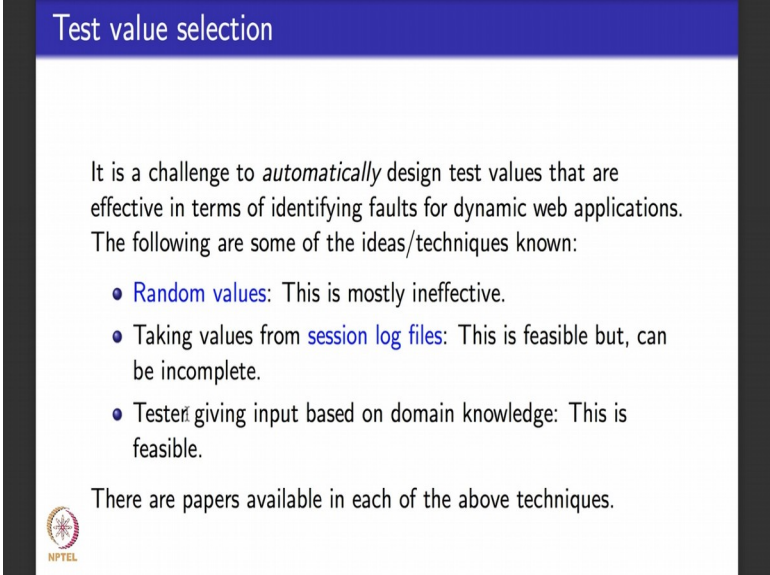
And we had broadly categorized web applications testing into 3 categories. Testing, static, hyper text web sites which I did as a part of my first lecture where we also discussed issues of testing web applications. Last class I taught you about client side testing of web applications in particular one technique called bypass testing. What I want to focus on in this lecture is server side testing of web applications. We will try to derive 2 specific graph models that correspond to server side testing, and I will tell you what those represent as far as testing the web applications software that present on the server that is concerned.

After this we will move on and do testing of object oriented applications.

So, I will skip past this slides that we have already discussed in the context of testing web applications. When we did dynamic testing I told you 2 categories, client side and server side. This is what you saw last time, client side testing, which basically involved

bypass testing is the technique that we learnt. I will move on to server side testing, this is where we had stopped last time.

(Refer Slide Time: 01:25)




### Test value selection

It is a challenge to *automatically* design test values that are effective in terms of identifying faults for dynamic web applications. The following are some of the ideas/techniques known:

- **Random values:** This is mostly ineffective.
- Taking values from **session log files:** This is feasible but, can be incomplete.
- **Tester giving input based on domain knowledge:** This is feasible.

There are papers available in each of the above techniques.



We talked about how to select test cases when we do server, server side testing. Just to quickly recap we are doing system level testing of web applications. When I do a method level testing or I do a component level testing then whatever we discussed in the traditional ways like, graph based logic based, mutation testing they all can be applied.

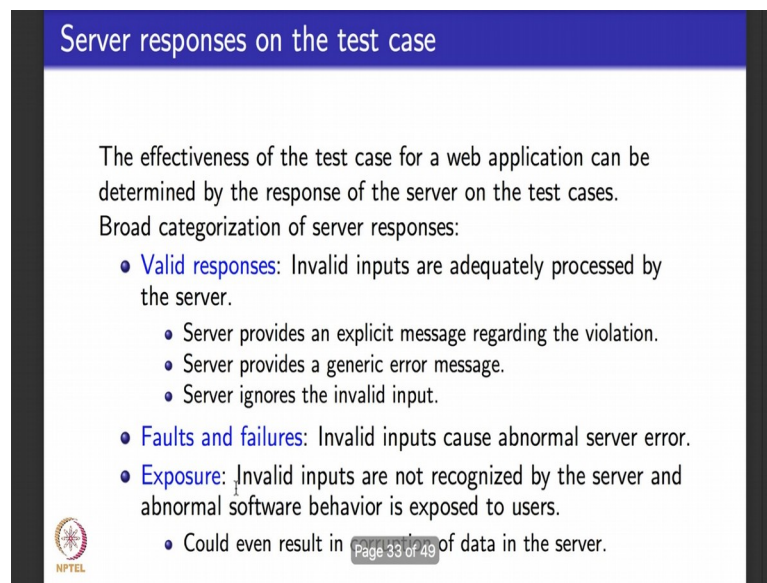
I am assuming that all that is done now. I have the full code integrated into the server with the clients available and I am doing system level testing which comes towards the end of testing phase. And I want to focus on system level testing of web applications on the server side. When it comes to that, when we say test values on the server side, what is the big challenge? The idea is to automatically design test cases or test values that are effective in terms of finding faults. For what kind of web applications, for dynamic web applications, that is what we are concentrating on.

This is just recapping what we discussed. One option is to select random test cases and test. This is useful sometimes, but not specifically effective all the time. The other idea is to take values from session log files. I will not be discussing this technique in detail, but I will point you to a reference material at the end of this lecture, where you can get some more information about how to test based on user session data. The third option is to get

for the tester to give inputs based on domain knowledge. One technique that we already discussed in this category was bypass testing, which we did on the client side.

One more technique that we will discuss for the server side based on this, is what we are going to discuss today which will involve deriving graph models from the server side. And as I told you I will give you pointers to papers where you can know in detail about all these techniques.


(Refer Slide Time: 03:20)



**Server responses on the test case**

The effectiveness of the test case for a web application can be determined by the response of the server on the test cases.

Broad categorization of server responses:

- **Valid responses:** Invalid inputs are adequately processed by the server.
  - Server provides an explicit message regarding the violation.
  - Server provides a generic error message.
  - Server ignores the invalid input.
- **Faults and failures:** Invalid inputs cause abnormal server error.
- **Exposure:** Invalid inputs are not recognized by the server and abnormal software behavior is exposed to users.
  - Could even result in  of data in the server.

Page 33 of 49

Before we move on and discuss server side testing, when we say I want to define test cases to do system level testing that will pull out errors, what could be errors related to server side testing of web application? So, errors could be handled by the server, they could be detected and handled by the server, they could be detected and not handled by the server in the sense that they cause an error.

In addition to that, not only causing an error it could result in permanent damage to the server in the sense of corrupting the data on the data base that is present in the server and so on. So, server when it handles the test case the error caused by the effective test case, server could produce a valid response. What is a valid response? A valid response is any response that is produced by the server in response to handling an invalid input that is generated by a test case. Servers aware of the fact that the input is invalid and it responds in one of the following way. It responds by providing an explicit message saying that

there is a problem with this input, this input is not of this format and it explains the violation.

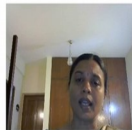

It could respond by providing a generic error message which is not fine tune to the kind of violation that happens or server could just ignore the invalid input. What are we mean by ignoring the invalid input? It recognizes the fact that it is an invalid input, ignores in the sense of does not intimate the tester or the client that the input is invalid, ignores it goes ahead and does its job normally. The second categorization of server responses could result in a fault or a failure. A fault is an internal fault, it may or may not propagate to being visible in the output. A failure propagates to being visible in the output. Both the cases there is a server error and it is abnormal in nature.

And when the failure gets exposed what happens is that the invalid inputs that are not recognized by the server, the user also gets to know that it is not recognized by the server, because there is a visible exposure of that error from the information that the server provides to the user. Or the server may not provide any information to the user. In fact, a very worse situation can happen which is the invalid input is so bad that the server does not only produce abnormal errors, it results in corruption of data in this server like I told you it could corrupt the data present in the data base in the server fix sometimes could cause more damage than intended.

(Refer Slide Time: 06:02)

### Server-side (white-box) testing

- If server-side source code is available, we can use our usual graph models to test the server.
- Control flow graph exhibits only *static* models, not effective for web applications.
- **Presentation layer** of a web application contains the software and is useful to do testing.
- For software in the presentation layer, two graph models exist:
  - **Component Interaction Model (CIM)**
  - **Application Transition Graph (ATG)**



So, when it comes to server side white box testing what are we going to do? If the source code of the server is available as I told you to do method level testing, unit level testing, we can use our usual graph models to test the server. But the problem is for system level testing of server, graph models like control flow graphs, do not really tell you about the dynamic behavior of the server. They are static models that describe the structure of the code that is return for the server, they do not talk about how this server put together with the client behaves as a system that is put together.

So, what we want to do is we want to derive and work with 2 new graph models that talk about the system level behavior from the server side. When we talk about the server side code you remember 2 classes ago, I presented to you the architecture of web application which involves several different layers, there was a layer containing the data then there was a layer that contains, it is user interface. In between these 2 layers there was a presentation layer. What we are focusing on is the code that is present that corresponds to the server as it is given in the presentation layer. From this code that is present in the presentation layer of architecture of a web application we are going to derive 2 different graph models.

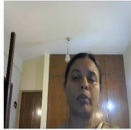

The first graph model that we are going to work with is called component interaction model, abbreviated as CIM. The second graph model that we are going to work with is called application transition graph model abbreviated as ATG. I will first begin with the CIM model then move on to ATG model.

(Refer Slide Time: 07:43)

### Atomic sections

We need the notion of **atomic sections** to understand the graph models.

- An **atomic section** is a section of HTML with the property that if any part of the section is sent to a client, the entire section is.
  - May include JavaScript
  - All or nothing property
- A HTML file is an atomic section.
- A **content variable** is a program variable that provides data to an atomic section.
- Atomic sections may be empty.




So as I told you please remember we looking at the server side code as it is present in the presentation layer of a web application that is dynamic in nature. So, when it is dynamic in nature, a particular website can be created in different ways in respond to different requests.

And it is created by dynamically generating HTML forms and when HTML forms are dynamically generated a part which generated in sections of fragments. And there are sections of fragments that are specifically called atomic sections that are very important for us. So, what is an atomic section? An atomic section is a section of a HTML file with the following property, if one part of the section is sent to the client then the entire part of the section is sent. Before moving on I will take an example and explain to you what it is.

(Refer Slide Time: 08:32)

### Atomic sections: Example

	PrintWriter out = response.getWriter();
P1 =	out.println("<HTML>") out.println("<HEAD><TITLE>" + title + "</TITLE></HEAD>") out.println("<BODY>")
	if(User) {
P2 =	out.println("<CENTER> Welcome!</CENTER>"); for (int i=0; i<myVector.size(); i++) if (myVector.elementAt(i).size > 10)
P3 =	out.println("<p><b>" + myVector.elementAt(i) + "</b></p>"); else out.println("<p>" + myVector.elementAt(i) + "</p>");
P4 =	} else
P5 =	{ }
P6 =	out.println("</BODY></HTML>"); out.close ();



Atomic sections are coloured in red. P1, P2, P3 are empty atomic section. Content variables are coloured green.

Page 36 of 49

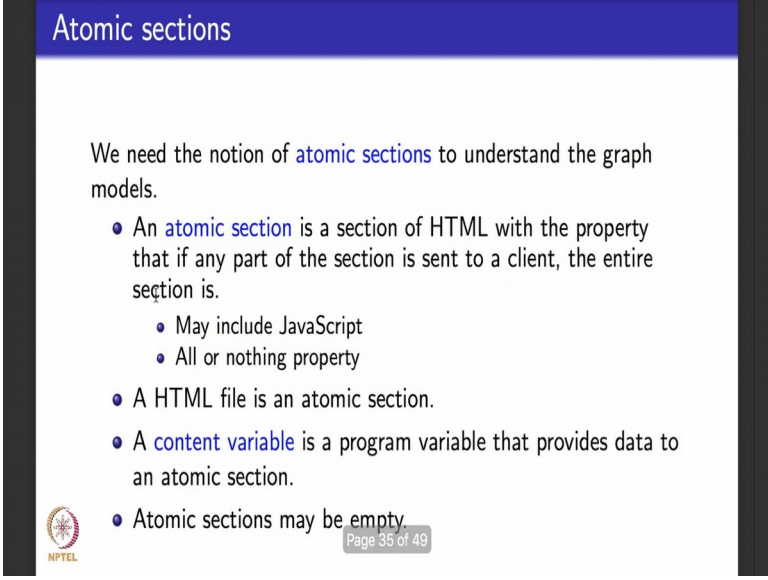
So, here is how the code looks like, the this thing that I bracketed as different rows in the table corresponds to some server side code present in the presentation layer.

So, it first gets a response from the client, and then it prints out this chunk. First remember it is to print out a HTML file to be rendered in the web browser interface of the client. And every HTML file begins with the HTML tag followed by a header tag followed by a body tag. So, I have put all these 3 print statement that print the begin of the HTML header tag, the HTML main tag followed by the header tag, which prints the title and ends the head tag followed by the beginning of the body tag. All these 3 are printed together.

So, they are put as one row in the table and constitute one atomic section. After that if the user is authenticated. If the user is a valid user then a welcome message is printed and input is being read. If the user is an invalid user go down here which is the else part then nothing is printed, and the HTML file is ended and closed. So, an atomic section is a chunk of HTML file such that if one part of the file is output to the client, is sent for printing by the client at the end, then the entire part is. Here this is one chunk that is printed this is the next chunk that reads the, if the users validated prints the message, this is the next chunk that is printed. So, this is the next chunk which is empty.




(Refer Slide Time: 10:14)



### Atomic sections

We need the notion of **atomic sections** to understand the graph models.

- An **atomic section** is a section of HTML with the property that if any part of the section is sent to a client, the entire section is.
  - May include JavaScript
  - All or nothing property
- A HTML file is an atomic section.
- A **content variable** is a program variable that provides data to an atomic section.
- Atomic sections may be empty.

 Page 35 of 49

So, that is what is described here and atomic section is a section of the HTML file with the property that if any part of the section of the HTML file is sent to the client the entire section is. Atomic sections might themselves include scripts return in Java script. It could be empty, atomic sections basically satisfy all or nothing. If the thing is printed it is printed in its entirety or nothing is printed. One full HTML file could be printed as an atomic section. Atomic sections could be empty. Another related terminology that we will need is the notion of a content variable. What is the content variable? It is a program variable that provides data to an atomic section.

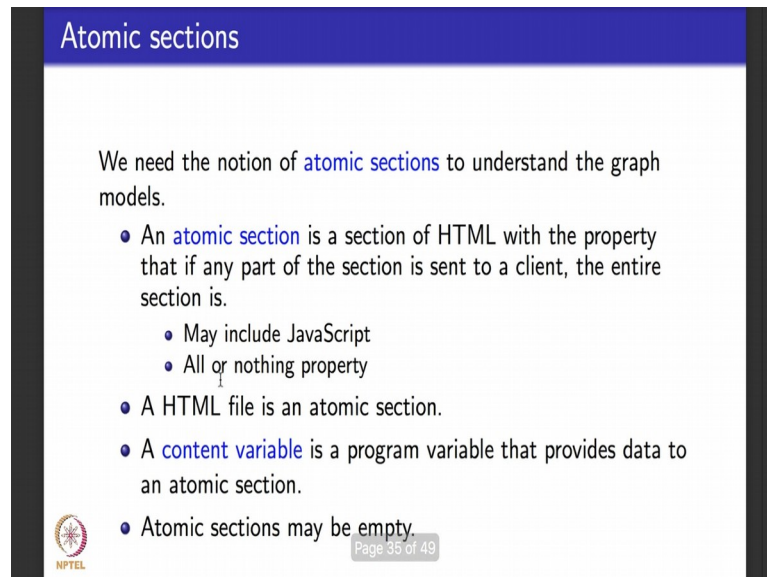
So now if you go in this example these things that I have bifurcated by using these lines to see how the code is fragmented, the ones that are marked in red on this column on the left hand side represent atomic sections. So, there are 6 different atomic sections. If you notice there is one atomic section P5 which is completely empty. In this scenario, why is it empty? It is empty because the user is not been validated. So, they do not want to print the welcome message and read the data and print it. So, they print nothing, but they close the body and HTML file so that the empty HTML file can be rendered back on the client's web browser. Content variables as we see are colored in green.

So, here title is a content variable. What is a content variable? That generates data dynamically. So, title of the web page is dynamically generated. For example, let us say if bank page it could print welcome message followed by your name as a part of the title.



And the name part of the welcome message is dynamically generated. This data, whatever it is which is stored in this vector element, it could be data related to your statement if it is a bank page it could be data related to your marks, if it is your marks page it could be anything this data that is dynamically generated is also called content.

(Refer Slide Time: 12:17)



### Atomic sections

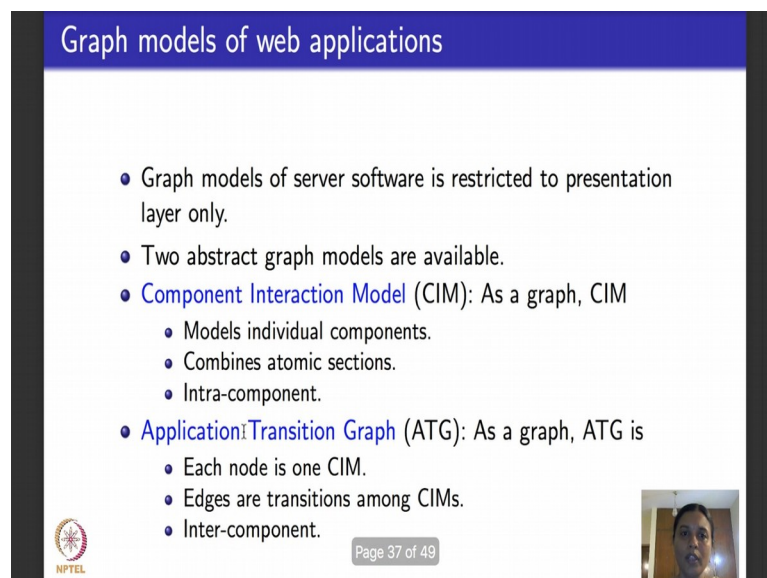
We need the notion of **atomic sections** to understand the graph models.

- An **atomic section** is a section of HTML with the property that if any part of the section is sent to a client, the entire section is.
  - May include JavaScript
  - All or nothing property
- A HTML file is an atomic section.
- A **content variable** is a program variable that provides data to an atomic section.
- Atomic sections may be empty.

NPTEL Page 35 of 49

So, I hope these 2 terminologies are clear, atomic section and content variable.

(Refer Slide Time: 12:21)



### Graph models of web applications

- Graph models of server software is restricted to presentation layer only.
- Two abstract graph models are available.
- **Component Interaction Model (CIM)**: As a graph, CIM
  - Models individual components.
  - Combines atomic sections.
  - Intra-component.
- **Application Transition Graph (ATG)**: As a graph, ATG is
  - Each node is one CIM.
  - Edges are transitions among CIMs.
  - Inter-component.

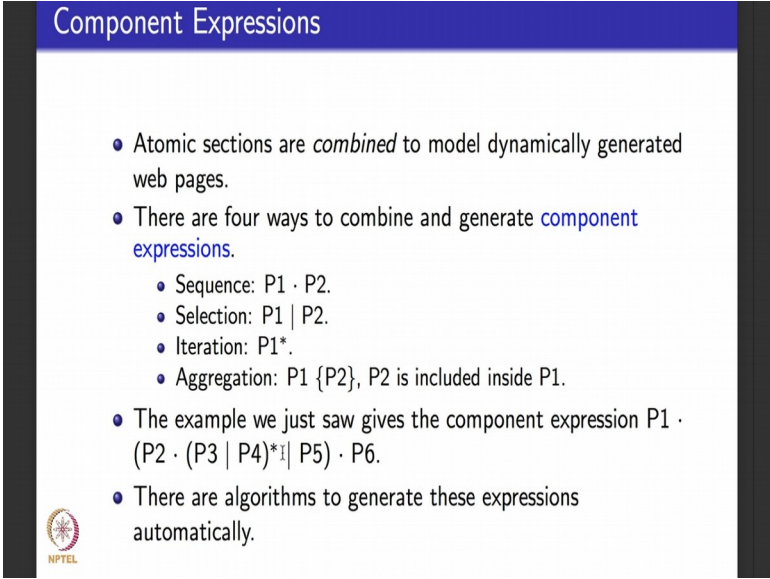
NPTEL Page 37 of 49

Moving on, our 2 graph models that we will be considering will be built on atomic sections. And then further on the first level graph model. So, component interaction

models CIM as a graph, what does it do? Models individual components combines atomic sections. It is intra component means it talks about interaction within a component. Nodes of a component interaction model graph will be atomic sections; transitions will tell you how to move from one section to the other, with that thing it will model one particular component of a web server software that is present in the presentation layer.


Building on the CIM model, we will later on define the application transition graph or the ATG model where each node of the ATG model will be one CIM graph and edges would be transitions amongst the CIM graph. Because each node is one CIM graph and application transition graph is a inter component graph, that is it models interactions across components. I will first begin with CIM.

(Refer Slide Time: 13:30)



**Component Expressions**

- Atomic sections are *combined* to model dynamically generated web pages.
- There are four ways to combine and generate **component expressions**.
  - Sequence:  $P1 \cdot P2$ .
  - Selection:  $P1 \mid P2$ .
  - Iteration:  $P1^*$ .
  - Aggregation:  $P1 \{P2\}$ , P2 is included inside P1.
- The example we just saw gives the component expression  $P1 \cdot (P2 \cdot (P3 \mid P4)^* \mid P5) \cdot P6$ .
- There are algorithms to generate these expressions automatically.

 NPTEL

So, in before we understand CIM we need to know what component expressions are. Atomic sections which we saw little while ago, they can be combined to model dynamically generated web pages. How will you combine atomic sections? If you remember when we did mutation testing, I had introduced you to regular expressions. Regular expressions is one standard way of combining atomic sections.

So, this sequential operator, this is P1 is one atomic section. P2 is one atomic section.  $P1.P2$  means atomic section P1 is followed by atomic section P2 in a sequential way in this order. This is a same syntax as that of sequential concatenation in a regular

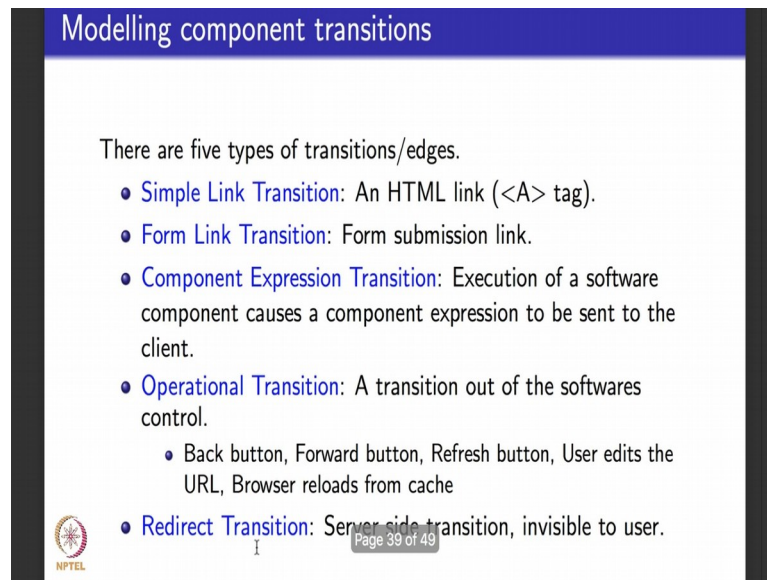
expression. Selection which corresponded to + as we defined it in regular expression is union which means either print P1 or print P2, print one of P1 or P2, iteration is again very similar to the regular expression iteration print 1, 0 or more occurrences of the atomic section P1. In addition to these three standard regular expression like operators we also have the operator of aggregator.

What is aggregator do? If the syntax is like this: it is says P1 and then you open these flower braces or curly braces and then you print P2. So, the way to read it is first P1 is printed as an atomic section. And inside the atomic section P1, P2 is printed right. So, these are the 4 ways to combine component expressions. If you go back to this example that we saw, atomic section P1 is printed once, this begins the HTML file, and then this is a decision statement which corresponds to the internal code present in the server. If this decision is positive, turns out to be true, which means the user is validated then atomic section P2 is printed.

Where P2 begins with the welcome message, then what does this one do? This one goes through the input the data containing, the input is present in this array called my vector. So, this is a for loop that goes through that, and as it is going through the thing in different paragraph, in a new line break it prints each element as it is reading from the input, and it keeps doing that and when it ends it exits and comes out. So, this is printed as many times as there are inputs to be read. So, the sequential composition of that thing would be you print P1 and then you do P2 and then you either print P3 or P4 as many times as the for loop would run or if the user is not validated print the empty atomic section P5.

Whatever you do, follow it with P6 which closes the HTML body and main HTML type. So, this is how atomic section are combined, and this is the graph also corresponding to the component interaction model. Given a particular web server code there are algorithms to generate these expressions automatically as I told you I will give you references from where you can get these algorithms to do it.


(Refer Slide Time: 16:48)



**Modelling component transitions**

There are five types of transitions/edges.

- **Simple Link Transition:** An HTML link (<A> tag).
- **Form Link Transition:** Form submission link.
- **Component Expression Transition:** Execution of a software component causes a component expression to be sent to the client.
- **Operational Transition:** A transition out of the softwares control.
  - Back button, Forward button, Refresh button, User edits the URL, Browser reloads from cache
- **Redirect Transition:** Server side transition, invisible to user.

 Page 39 of 49

So now, what are the various kinds of transitions that can occur in a component interaction model? There are 5 different kinds of transitions, a transition could be the result of a simple HTML link.

They anchor tag which many of you might know is one such example, I just in a part of a HTML text, I just use the anchor tag and hyper link to another page. So, this is an example of a static link or a simple link. The next example of a transition could be a form link transition, which is basically you typing some data like your user ID and password and you are exposed to a form from which you type in this data and you submit. When you submit you create what is called a form link. The third kind of link could be a component expression transition which is particular execution of a particular software component causes one whole expression to be sent to the client.

An expression like this like the one that we saw earlier. The fourth could be an operational transition, what is this mean? This means that is the sudden transition that comes as an a unplanned input from the user. Typically if you are accessing your bank side or if you are doing your online shopping for some reason and you have typed in your credentials and you are accessing your bank's site, you will often find the information which says do not press back or refresh button, why? Because the particular data is being encrypted and sent across from the client which is a instance of your

machine from which you are accessing your bank to the server, this takes a while for a server to process that data validate it and print it.


And it might if you run out of patients and press back button then you interrupt to the processing of the server. And those kind of things are called operational transitions. It is a transitions that was not as a part of the planned routine that disrupts the operations that the server perform some particular client input data. Typically by pressing back button, forward button refresh button or you change the URL browser reloads from cache, all these things cause abnormal operational transitions. The other thing could be what is called redirect transition which is an internal transition done by the server.

It is typically not visible to a user who sitting on the client. To give you an example of what are redirect transition could be like for example, the server code could be modularized and for the server to be able to particularly, let us say it has to need, it has to fetch some kind of data from a data base which is sitting in another server, and let us say the main web server has access to the data base server this main web server could do a redirect transition to the data base server, get the data or validate the data and accordingly respond. So, that kind of transitions have called redirect transitions. So, transitions that are simple hyper links, transition that of forms, transitions that corresponds to component expressions, transitions that are abnormal which cause abnormal interruption in server processing of data and these are transitions that are internal to the server.

(Refer Slide Time: 19:58)

CIM Example: gradeServlet

	<pre>ID = request.getParameter ("Id"); passWord = request.getParameter ("Password"); retry = request.getParameter ("Retry"); PrintWriter out = response.getWriter();</pre>
P1 =	<pre>out.println ("&lt;HTML&gt;&lt;HEAD&gt;&lt;TITLE&gt;" +title+ "&lt;/TITLE&gt;&lt;/HEAD&gt;&lt;BODY&gt;") if ((Validate (ID, passWord)) {</pre>
P2 =	<pre>out.println ("&lt;B&gt; Grade Report &lt;/B&gt;"); for (int l=0; l &lt; numberOfCourse; l++)</pre>
P3 =	<pre>out.println("&lt;p&gt;&lt;b&gt;" +courseName(l)+ "&lt;/b&gt;" + courseGrade(l)+ "&lt;/p&gt;"); } else if (retry &lt; 3) {</pre>
P4 =	<pre>retry++; out.println("Wrong ID or wrong password"); out.println("&lt;FORM Method='get' Action='gradeServlet'&gt;"); out.println("&lt;INPUT Type='text' Name='Id' Size=10&gt;"); out.println("&lt;INPUT Type='password' Name='Password' Width=20&gt;"); out.println("&lt;INPUT Type='hidden' Name='Retry' Value='"+(retry)+"&gt;"); out.println("&lt;INPUT Type='submit' Name='Submit' Value='submit'&gt;"); out.println("&lt;a href='sendMail'&gt;Send mail to the professor&lt;A&gt;");</pre>



So, how does a CIM model look like? We will expand on what we did in this model here. I just told you here for my vector element. Let us say this is something that prints the marks of a particular student. A student is waiting for results let us say of his tenth standard or twelfth standard or GATE exam. Results are going to be made available online on a particular date and if the user is valid, this actually reads the data of the score or the marks corresponding to the subject and prints it out.

So, what I will do is I will retain some parts of it the later parts of this, but I will elaborate on this part of it. So, this is an example of a GradeServlet, grade in the sense of a student grades. So, you start by saying you request for ID request for password. If the password is wrong then you request for retry and then you get a response. And then you start printing, this is the header tag, this is the body tag. If the ID and password are correct then you start printing the grade report. This is the title and let us say some student might have written 5 courses, 6 courses, 4 courses, different number of courses. So, you run a loop through the number of courses print the course name and the grade.


If the password is wrong then you are allow 3 attempts to correct your password. So, in the retry is the variable that captures the number of attempts. If retry is still less than 3 then you say your password or ID was wrong. And again redo this thing of password retries, submit and update retry here. And when the number of attempts cross then you say I have sent a mail to professor indicating that this is a problem. So, is it clear? And

then towards after this P4, I ended with this same P5 and P6 which I have not repeated which is empty, P6 closes the HTML tag. So, let us say this is an example of a code on the server which prints the grades.

(Refer Slide Time: 21:53)

CIM for gradeServlet

- Start page S is login.html.
- $A = \{P1, P2, P3, P4, P5, P6\}$ .
- $CE = \text{gradeServlet} = P1 \cdot ((P2 \cdot P3^*) \mid P4 \mid P5) \cdot P6$ .
- Transitions  $T = \{\text{login.html} \rightarrow \text{gradeServlet}[\text{get}, (\text{Id}, \text{Password}, \text{Retry})], \text{gradeServlet.P4} \rightarrow \text{sendMail}[\text{get}, ()], \text{gradeServlet.P4} \rightarrow \text{gradeServlet}[\text{get}, (\text{Retry})]\}$



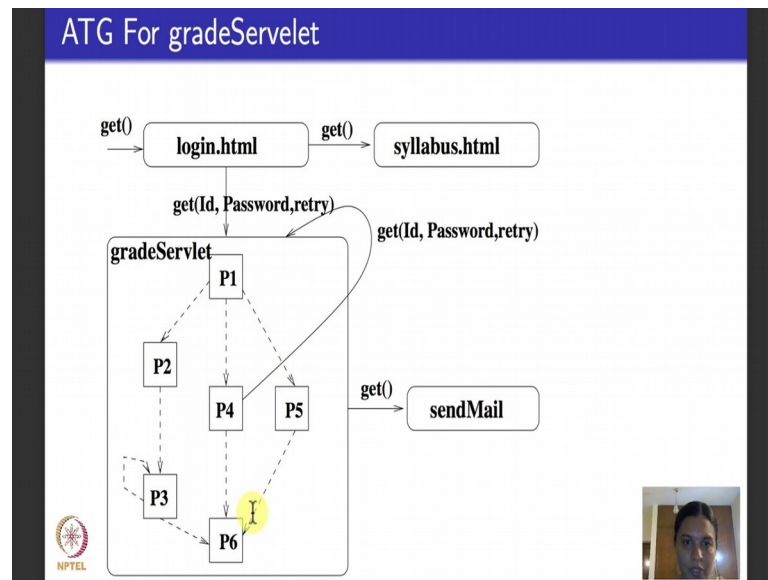
Page 41 of 49

I

So now let us generate the component interaction graph model for this. Start page will be login dot HTML which is not described here. And then atomic sections will be P1, P2, P3, P4 which are these, P5 and P6 generated from the earlier side and like we did last time always do P1 which is the header information. And then do P2 which is validating and print P3 as long as there is code, our P4 which is retry P5 retries crossed, ended with P6 which is the final ending of HTML tag. So, what are the transitions from login dot HTML I go into the grade servlet which is this component this entire component and inside grade servlet I can do transitions from P4 which talks about sending mail or we talk about number of retries of a password and so on. This is that part.

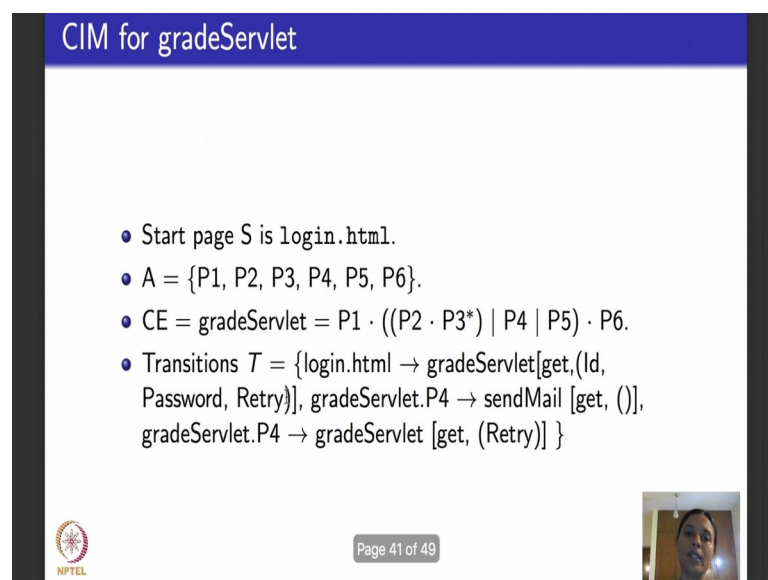


(Refer Slide Time: 22:50)



Just concentrate only on this where I am pointing my mouse to, P1 could be followed by P2 any attempts of P3 or P6 or P1 could be followed by P4 followed by P6 which corresponds to password retrials up to 3 times, or P1 could be followed by P5 could be followed by P6 which corresponds to failure after the 3 retrials and empty page is printed.

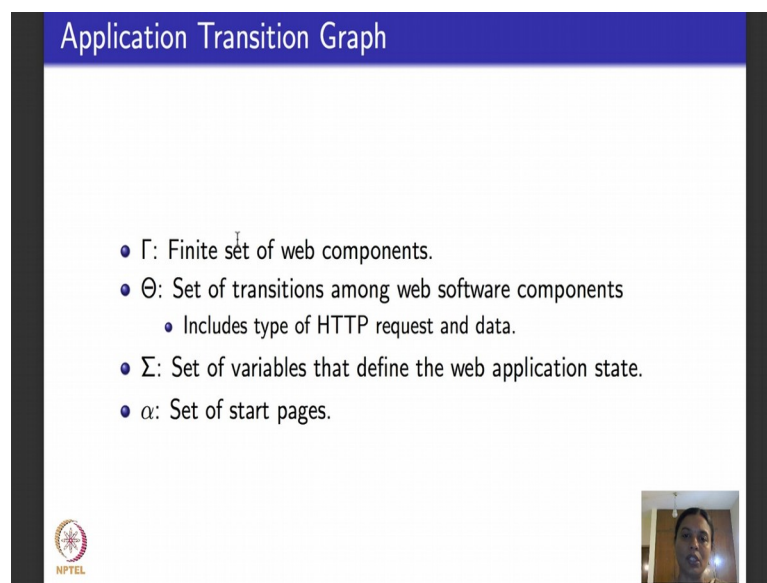
(Refer Slide Time: 23:16)



So, this is how a CIM looks like. I hope this example explain to you to a certain extent how a CIM model looks like.

Now, moving on we will look at how an application transition graph model looks like. If you go back to the slide where I introduced these 2 models to you, I said an application transition graph is also a graph where each vertex or a node is one CIM graph and edges are transitions amongst the components. So, it models inter component behavior at the presentation layer. Let us say to define an application transition graph, let us say we have a finite set of web components called  $\Gamma$ ,  $\Theta$  is the set of transitions amongst the web software components.

(Refer Slide Time: 23:50)

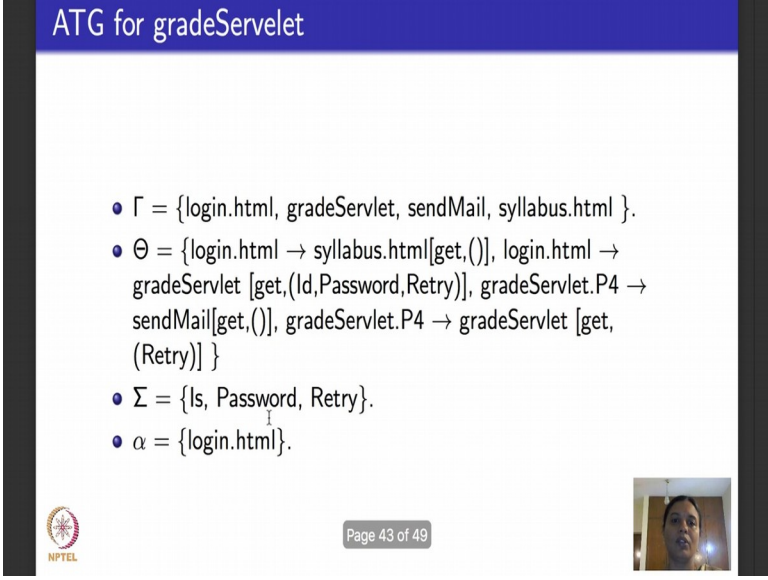


The slide is titled "Application Transition Graph" in a blue header. It contains a bulleted list of definitions for the components of the graph. In the bottom left corner, there is an NPTEL logo. In the bottom right corner, there is a small video inset showing a person's face.

- $\Gamma$ : Finite set of web components.
- $\Theta$ : Set of transitions among web software components
  - Includes type of HTTP request and data.
- $\Sigma$ : Set of variables that define the web application state.
- $\alpha$ : Set of start pages.

This include this could include HTTP requests sending of data, abnormal button presses any kind of transitions that we discussed earlier. And  $\Sigma$  is the set of content variables that defines this state of a web application and  $\alpha$  is the set of start pages.

(Refer Slide Time: 24:17)




ATG for gradeServlet

- $\Gamma = \{\text{login.html}, \text{gradeServlet}, \text{sendMail}, \text{syllabus.html}\}.$
- $\Theta = \{\text{login.html} \rightarrow \text{syllabus.html}[\text{get},()], \text{login.html} \rightarrow \text{gradeServlet} [\text{get},(\text{Id},\text{Password},\text{Retry})], \text{gradeServlet.P4} \rightarrow \text{sendMail}[\text{get},()], \text{gradeServlet.P4} \rightarrow \text{gradeServlet} [\text{get},(\text{Retry})]\}$
- $\Sigma = \{\text{Id}, \text{Password}, \text{Retry}\}.$
- $\alpha = \{\text{login.html}\}.$

NPTEL

Page 43 of 49



So in the grade servlet example they could be 4 different web files login dot HTML whose code I didn't give you let us say it is a static website that just has a form, grade servlet whose code we saw send mail whose code I didn't give you, but let us say it is similar to the grade servlet in the sense that it fixes up the email ID from the database and sends an email and then there is another static page let us say syllabus dot HTML.

So, the transition could be from login page you could go to a page where you get the syllabus dot HTML, fetch your data or from the login page you could directly go to a page where you print the grade details provided ID password are valid and retries within the allowed amount or, and from the grade servlet page you could go to a page where the grade report is sent by email or, from the grade servlet page you can go back to the grade servlet page because one of the attempts to enter user ID and password failed and you are still retrying within the 3 attempts to get it right.

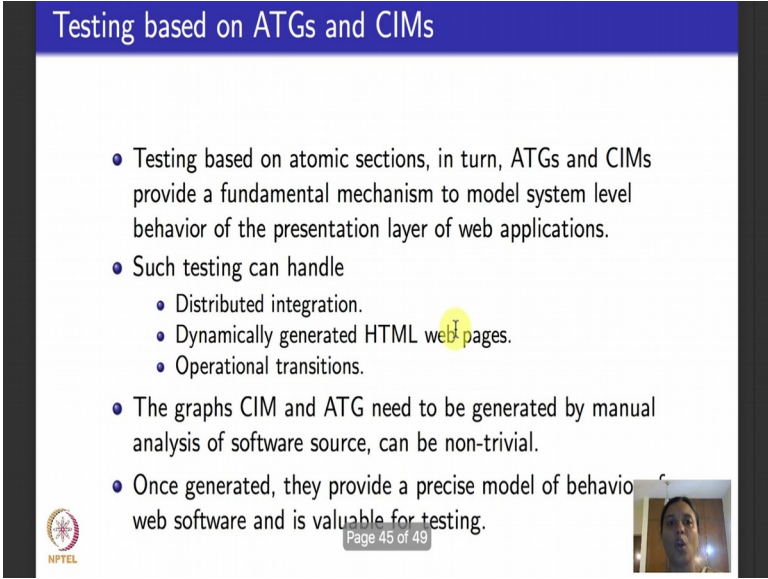
And what is this set of action set of actions or the variables which is defined are what is the ID sorry, there should be ID not IS, ID, ID name password and the number of retries the begin page is the login dot HTML page. So, here is how the ATG graph looks like. I go through a get request I go to the first page which is a login page, and then I can go to the syllabus page where I printed somebody might just want to know this syllabus you print it and you stop nothing more to do. From the login page if I get ID and password I

am allowed 3 retries as long as  $\text{retry} < 3$ . So, here is where I print the beginning thing if my ID and password is wrong, I go back and retry, this is presented lose ended here.

But the internal code server will ensure that this transition happens only 3 times. But if everything is correct then the grade card and the marks are printed here and you exit. The number of retries exceed 3, another ID and password still do not turn out to be right, then, you print the empty page and exit. Whatever it is you go to the send mail page. So, this whole entity describes the complete behavior of the server side software, as it is present in the presentation layer and inside this ATG, is one of the pages gradeservlet and one of the components gradeservlet.

We actually sort of zoomed in and looked at the CIM model which I discussed to you. Similarly you could have another CIM model here inside send mail, because this could also be a dynamically generated server code. But these this is probably static because it is just prints the syllabus, and this could probably have a form. So, if you kind of zoom in to each of these nodes, they could be similar internal graphs that correspond to the CIM graphs for each of this modules. So, this is how you generate graphs corresponding to system level dynamic behavior of a web software that is present in it is presentation layer.

(Refer Slide Time: 27:20)



The slide is titled "Testing based on ATGs and CIMs" in a blue header. It contains a bulleted list of four points. The first point states that testing based on atomic sections, in turn, ATGs and CIMs provide a fundamental mechanism to model system level behavior of the presentation layer of web applications. The second point states that such testing can handle three sub-points: distributed integration, dynamically generated HTML web pages, and operational transitions. The third point states that the graphs CIM and ATG need to be generated by manual analysis of software source, can be non-trivial. The fourth point states that once generated, they provide a precise model of behavior of web software and is valuable for testing. In the bottom left corner, there is an NPTEL logo. In the bottom right corner, there is a small video inset showing a person speaking. A small text box at the bottom center of the slide reads "Page 45 of 49".

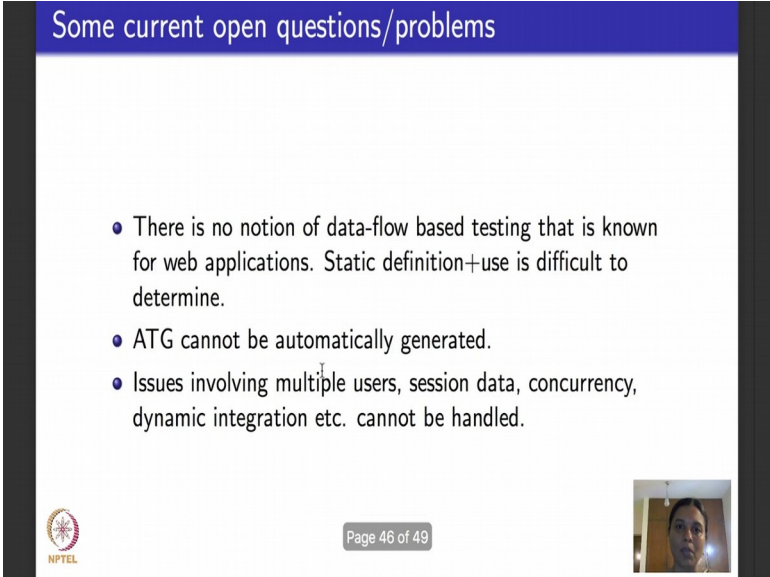
- Testing based on atomic sections, in turn, ATGs and CIMs provide a fundamental mechanism to model system level behavior of the presentation layer of web applications.
- Such testing can handle
  - Distributed integration.
  - Dynamically generated HTML web pages.
  - Operational transitions.
- The graphs CIM and ATG need to be generated by manual analysis of software source, can be non-trivial.
- Once generated, they provide a precise model of behavior of web software and is valuable for testing.

Once you generate this graph you can do whatever you want with it. You can do path testing, you can skip and test, you can generate test cases that correspond to identifying

various errors. So, testing based on atomic sections which in turn involves testing using ATGs and CIMs, it provides a nice mechanism to capture fundamental system level behavior of a server software. And please remember software of the presentation layer not the database layer. Such a testing, through examples, has proven to handle distributed integration of various websites, it can handle dynamically generated web pages to see if all the different ways of generating web pages can be done correctly.

It can handle operational transitions which are, which if you remember I told you were interrupts because somebody press the refresh button while a page was loading. Somebody press the back button while a page was loading, you can test for all these features of a server at the system level. The problem is that the graph CIM and ATG partly have to be generated manually. ATG graph, we do not know of an automatic way to generated by looking at the server side software. And this can be nontrivial, but once somebody is willing to putting that effort that in generate the graph then you give valuable information to test for all these properties of a web server, that is the point that is being made.

(Refer Slide Time: 28:42)

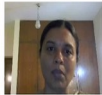


Some current open questions/problems

- There is no notion of data-flow based testing that is known for web applications. Static definition+use is difficult to determine.
- ATG cannot be automatically generated.
- Issues involving multiple users, session data, concurrency, dynamic integration etc. cannot be handled.

NPTEL

Page 46 of 49

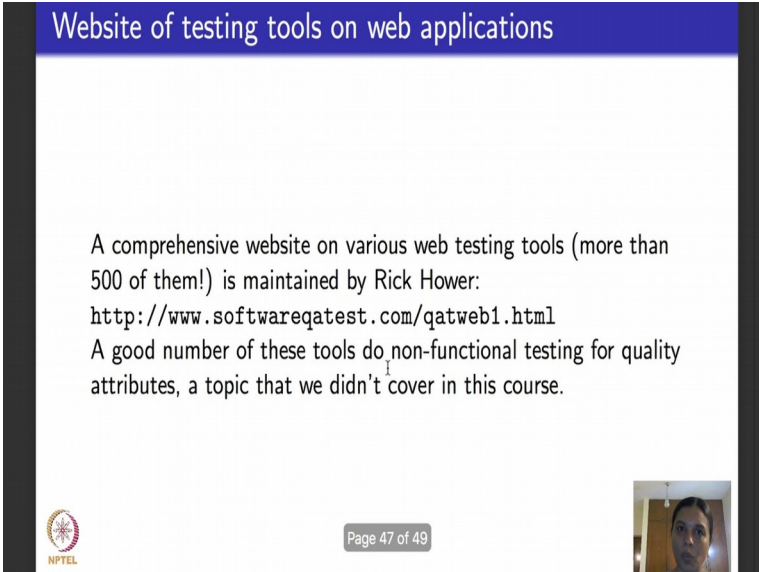


So, that is summarizes one technique for doing server side testing of web applications. So, just to recap what we did from the beginning, I introduced you to faults in websites. And then we looked at client side testing, one technique bypass testing. In this lecture I told you about graph based testing for system level testing of server code of web

software, what is left as far as web application testing is concerned. This area still very rich lot of open problems for example, if you consider what how does data flow graph model. If you see the ATG graph and CIM graph that we looked at models only control flow but at this system level, does not really talk about what happens to data flow. There is no clarity on data flow testing for system level web applications. People do not know how to do definition and use across dynamically generated web pages. So, that is very much an open problem that is left for working.

As I told you ATG cannot be generated automatically. Algorithms that will look at the server code on the presentation layer and generate ATG automatically would be very valuable. For example, if there are multiple uses if they session data this concurrency in the sense of multiple clients from different geographically located places handling, we still do not know how to generate graph models and test. All these are still rich in open problems that people could consider working on.

(Refer Slide Time: 30:10)



Website of testing tools on web applications

A comprehensive website on various web testing tools (more than 500 of them!) is maintained by Rick Hower:  
<http://www.softwareqatest.com/qatweb1.html>  
A good number of these tools do non-functional testing for quality attributes, a topic that we didn't cover in this course.

NPTEL

Page 47 of 49

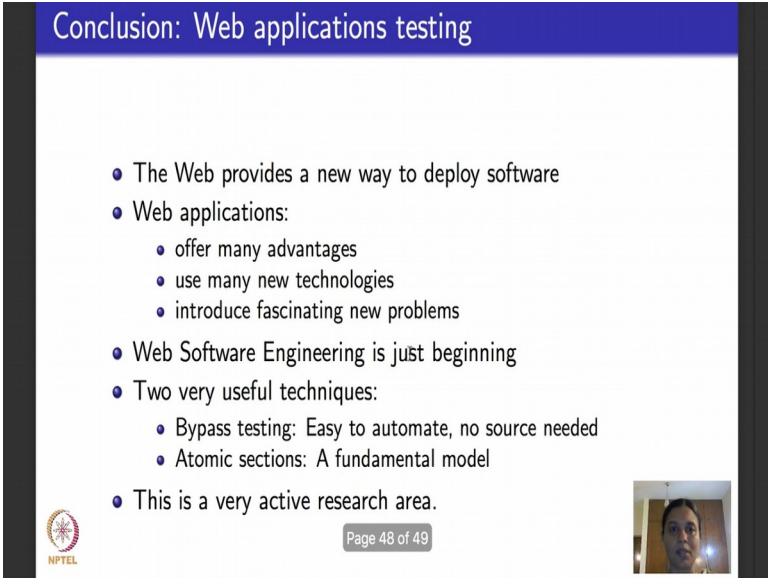
A small video inset in the bottom right corner shows a person's face.

Before I wind up this module on web applications testing, I would like to point to a nice website which maintains various web testing tools. In fact, if I know it write it as more than 500 different web testing tools. So, I just told you that there are several open problems and this website has almost 500 tools. So, you can imagine the richness of web application testing what we have done in these 3 lectures is probably to just scratch the surface of web application testing. This is the URL of this website, it is maintained this

by this gentlemen called Rick Hower. One thing to note before you access this website is that my focus on my lectures was on concentrating on system level functional testing of web application. I am testing the core functionality on the client side the first server side in terms of it does it do what it is supposed to do in terms of meeting it is functionality. That kind of techniques, testing techniques that I discussed with you does not involve things like performance testing, stress testing, low testing, which are non functional testing.

A lot of tools listed in this website talk about non functional testing of web applications. So, in case you are accessing this website, please remember that that something that we did not discuss in these 3 lectures on web applications.

(Refer Slide Time: 31:26)

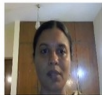


Conclusion: Web applications testing

- The Web provides a new way to deploy software
- Web applications:
  - offer many advantages
  - use many new technologies
  - introduce fascinating new problems
- Web Software Engineering is just beginning
- Two very useful techniques:
  - Bypass testing: Easy to automate, no source needed
  - Atomic sections: A fundamental model
- This is a very active research area.

NPTEL

Page 48 of 49



So, to conclude this module on web application testing. The web applications of a many advantages, they use several different technologies and as I told you there are still several fascinating open problems to work on. And I believe that is only the beginning of, if you want to coin a term called software engineering's specific to web software engineering. Research in it is still at it is beginning. What we looked at where 2 techniques bypass testing and graph models based on a atomic section. Bypass testing on the client side, atomic section on the server side, this is still a very active and rich area.





(Refer Slide Time: 32:04)

Some references

Here are some references for web applications testing.

- Sebastian Elbaum, Gregg Rothermel, Srikanth Karre and Marc Fisher II, Leveraging user-session data to support web application testing, in *IEEE Transactions on Software Engineering*, 31(3), 2005.
- Jeff Offutt, Ye Wu, Xiaochen Du and Hong Huang, Bypass testing of web applications, in *Proc. IEEE ISSRE*, 2004.
- Filippo Ricca and Paolo Tonella, Analysis and testing of web applications, in *Proc. ACM ICSE*, 25-34, 2004.
- Jeff Offutt and Ye Wu, Modeling presentation layer of web applications for testing, *Software and Systems Modelling*, 9(2), 257-280, 2010.



Page 49 of 49

So, here are some publications that I pulled out the material. From the first part which I did not do which is testing on the client side and on server side based on user session data. That you can find it in this paper by Elbaum, Rothermel, Karre and fisher it is an IEEE transactions on software engineering paper, about 12 years old. The thing that I taught you web pass testing bypass testing of web applications on the client side appeared in this paper by Offutt, Wu, Du and Huang in ISSRE 2004. This paper is another early paper that appeared in ICSE 2004 which talks about graph models of web testing the kind of things that I told you about 5 different transitions, between CIMs all those things you can find in this paper by Ricca and Tonella.

The server side testing that we discussed which is modeling the presentation layer as CIM and ATG. That was in the paper by Jeff Offutt and Wu which appeared in this general software and systems modeling in 2010. Feel free to pick up any of these papers if you want to know additional details or you could ask me in the forum if you have any specific questions if I note I will be able to answer it would be happy to do so. So in the next module will begin object oriented testing.

Thank you.