ICTSAS519 Perform systems tests – Part 2

Contents

2. Conduct Test

The previous resource looked at the various issues involved in planning to test a website. A modern dynamic website is a complex interaction of operating system, browser, web server, databases, scripting languages and networks. Because so many components are involved, it becomes virtually impossible to test every possible interaction between them. Undertaking careful risk analysis helps to determine where to focus our testing.

Once the preparation work is complete, what follows should be relatively easy. We need to execute our test plan carefully, ensuring that all of the tests are performed properly and results are noted carefully and accurately. Once the test is finished, we then need to analyse and interpret the results for stakeholders.

The topics contained within this resource are:

- Running Test Scripts
- Performing Benchmarks
- Industry Standards
- Checking Results
- Summary

2.1 Create clean test environment and initialise test environment according to test plan

In a previous module we looked at preparing a test plan for a website. When performing testing, it is important to have a plan in place to ensure all-important areas are covered and nothing is forgotten or left to chance. This ensures that testing is performed properly and that reasonable results can be drawn from the test session.

Usability Tests

There are many different techniques for usability testing and in many cases the type of usability testing you conduct on a website will depend on your budget. The two most commonly used techniques involve asking a test subject to perform a series of tasks using the website under test. With one method, you might place the test subject in front of a PC and observe what they do. The other form of usability test is conducted as an interview and might use a paper mock-up of the website.

You can quite often learn a lot about the usability of a website or application by asking people to use the site and observing how they go about it. If the test subject has trouble finding a particular option, it might mean that it's too small or worded badly. If they have trouble finding information on the site, it may mean you need to improve the search function or make it more prominent.

After conducting a usability test as discussed above it may be useful to interview a test subject to try and understand why they did things the way they did. However, usability guru Jakob Nielson warns against placing too much faith in the advice you receive from a test subject:

When talking about past behaviour, users self-reported data is typically three steps removed from the truth:

- In answering questions (particularly in a focus group), people bend the truth to be closer to what they think you want to hear or what's socially acceptable.
- In telling you what they do, people are really telling you what they remember doing. Human memory is very fallible, especially regarding the small details that are crucial for interface design. Users cannot remember some details at all, such as interface elements that they didn't see.
- In reporting what they do remember, people rationalize their behaviour. Countless times I have heard statements like "I would have seen the button if it had been bigger." Maybe. All we know is that the user didn't see the button.

(Nielson, J http://www.useit.com/alertbox/20010805.html)

Where the website involves an administrative interface that staff will use, you might choose to bring in representatives from the organisation to test the administrative interface. That way the staff that will be using the features daily can have some input into improving them.

In all cases though a developer of the website cannot do this sort of usability testing. A website tends to naturally match the developer's view of the requirements and how they should be implemented. In usability testing, we're essentially trying to ensure that the developer's idea about how a particular task should be accomplished is a natural fit with the actual users of the system.

The other form of usability testing involves a similar process. Again, we ask the user to perform a series of tasks, but this time we ask them to explain what they are doing and why. This is sometimes done using a paper copy of the website and asking people to point at what they want to click on and explain what they're doing. Using a paper model has several key advantages; firstly, it means that this form of usability test can be performed early in the development cycle before the website has actually been implemented. This sort of early testing can be used to validate your initial concepts before they are implemented. The earlier the need for change is identified the less costly the change will be.

Using a paper model also allows you to make immediate changes to the design. This sort of test can be useful for early validation of ideas. Usability testing requires good interview and listening techniques. Try using both open and closed questions in combination. An open-ended question is one with no fixed yes or no answer. Open-ended questions can be used to draw information out of your interview subject, which can be followed up later. Closed questions are those that lead to a simple yes or no answer. These can be useful for confirming your understanding of previous answers. Try to avoid using too many "Why ….?" questions, as these can sometimes make a user uncomfortable. "Why didn't you see the Checkout option?" is a good example of a question that should be avoided, as this seems to blame the interviewee.

A better technique might be to use something like the following:

Tester: After you selected the blue widget, you seemed to have some trouble purchasing it. Were you looking for a particular option?
Interviewee: I couldn't find something that would allow me to buy the items I had selected.
Tester: There is a "check out" option on the screen, did you try that?
Interviewee: No, I was looking for something like "purchase this", or "buy now".
Tester: Would it help if we changed the wording to "Purchase Item Now"?
Interviewee: Yes.

Here you can see that we used open ended questions to get a sense of the tester's problem, then followed up with a closed question to make sure we understand the problem and get some feedback on a proposed solution.

Proofing Content

Many different link-checking tools exist. A good link checker will usually process several hundred pages in minutes. Doing the same job manually would take hours and be somewhat more prone to error.

Let’s run a link checking program over a personal weblog site and see what it finds. The program we'll use for this example is a Linux command line utility called "linkchecker" (http://linkchecker.sourceforge.net).

Example: Link checker

```
[root@lithium root]# linkchecker -r3 http://www.example.com/
LinkChecker 1.12.3 Copyright © 2000-2004 Bastian Kleineidam
LinkChecker comes with ABSOLUTELY NO WARRANTY!
This is free software and you are welcome to redistribute it
under certain conditions. Look at the file `LICENSE' within this
distribution.
Get the newest version at http://linkchecker.sourceforge.net/
Write comments and bugs to calvin@users.sourceforge.net

Start checking at 2004-07-07 13:18:23+010

URL 000318010f86sabre.jpg
Name 000318010f86sabre.jpg
Parent URL http://www.example.com/photos/jetaction/Index.html, line 50,
col 36
Real URL
http://www.example.com/photos/jetaction/000318010f86sabre.jpg
Check Time 0.406 seconds
Result Error: 404 Not Found

URL http://www.leppo.com/~airplanehome/
Name The Boeing 727-200 Home Conversion Project
Parent URL http://www.example.com/old/main.htm, line 70, col 85
Real URL http://www.leppo.com/%7Eairplanehome/
Check Time 2.596 seconds
Result Error: 404 Not Found

That’s it. 2 errors in 2578 links found
Stopped checking at 2004-07-07 13:20:27+010 ( 2.068 minutes)
```

In the example above the linkchecker has been configured to check each page on the nominated site for broken links ("-r3" means "check this page and any pages within three links of this page"). It then returns the details of any broken links found, including:

- The file/site that wasn't found
- The page which contained the link and the line number containing the link

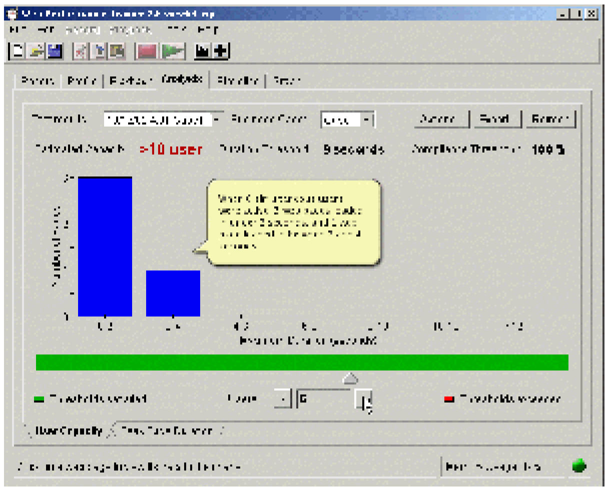
The sample output shows that an image on the site itself could not be found; this could be an error with the content uploaded to the site and perhaps the image wasn't uploaded correctly. The other error shows that an external site no longer exists; there's not much we can do about this. We could link to a copy of the site from the Wayback machine (http://www.archive.org/web/web.php), or remove the link altogether. With a tool such as this it becomes trivial to locate bad links or files missing from your website. Armed with this sort of information we can then fix the links.

2.2 Run test scripts and document results according to organisational testing and acceptance processes and test plan

There is no way of knowing in advance how much traffic a website might receive. Through historical data such as web server logs we can however make a reasonable guess about the number of visitors that can be expected. Load testing can demonstrate that the website can deal with the number of visitors expected with capacity to spare. For some projects a specific response time may form part of the specification and this should be demonstrated also.

There are quite a number of load testing tools available. These range from simple tools like ApacheBench which request the same page many times, to more complex tools such as Web Performance Trainer from Web Performance Inc. The more elaborate tools can be trained to mimic real users and may provide some more in-depth analysis of the data they collect.

Using the more elaborate tools allows us to build test cases which mimic real users, allowing us to load test websites in an extremely realistic fashion. Some tools may provide data analysis features. The screen shot below shows one of the reports generated by Web Performance Trainer. These tools can be extremely useful when conducting in depth load testing.



[click on image for a larger view]

ApacheBench is bundled as part of the Apache httpd server, but can be used to test any web server. There is a web gateway for ApacheBench but it is more commonly used as a command line tool. The example below shows the results from running ApacheBench on a Linux box to test the performance of the local web server.

Example: Apache Bench

```
[root@webserver root]# ab -n 1000 -c 25
http://my.webserver.net.au/index.php
This is ApacheBench, Version 2.0.40-dev <$Revision: 1.121.2.4 $>
apache-2.0
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Copyright (c) 1998-2002 The Apache Software Foundation,
http://www.apache.org/

Benchmarking my.webserver.net.au (be patient)
Send request timed out!
Send request timed out!
Send request timed out!
Completed 100 requests
Completed 200 requests
Send request timed out!
Send request timed out!
Send request timed out!
Send request timed out!
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Finished 1000 requests


Server Software: Apache/2.0.48
Server Hostname: my.webserver.net.au
Server Port: 80

Document Path: /index.php
Document Length: 1174 bytes

Concurrency Level: 25
Time taken for tests: 74.233838 seconds
Complete requests: 1000
Failed requests: 204
(Connect: 0, Length: 4, Exceptions: 200)
Write errors: 0
```

```
Total transferred: 1661436 bytes
HTML transferred: 1405278 bytes
Requests per second: 13.47 [#/sec] (mean)
Time per request: 1855.846 [ms] (mean)
Time per request: 74.234 [ms] (mean, across all concurrent requests)
Transfer rate: 21.85 [Kbytes/sec] received

Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 13 44.4 0 306
Processing: 101 589 3107.5 253 30280
Waiting: -288 0 14.8 0 0
Total: 103 603 3106.1 254 30280

Percentage of the requests served within a certain time (ms)
50% 254
66% 257
75% 275
80% 285
90% 322
95% 459
98% 799
99% 29936
100% 30280 (longest request)
```

The command used to start ApacheBench instructs requests http://my.webserver.net.au/index.php 1000 times (-n 1000), simulating 25 simultaneous or concurrent users (-c 25).

After displaying an initial copyright message and performing the tests, ApacheBench then displays a bit of information about the page under test. In the 74.2 seconds it took to request the page 1000 times, there were 204 errors and 796 successful requests. We can also see the number of requests handled per second and an average time per request. The number of requests per second indicates how many times per second this page can be served on the current hardware, this can then be compared to expected loads. The table at the bottom shows the percentage of requests and the number of milliseconds in which that percentage were served. This tells us that half of the requests were served in 254ms or less, 2/3rds of the requests were served in 257ms or less and 95% of requests were served in

459ms or faster. This data can be useful in situations where we're required to demonstrate that the website remains responsive under load.

Overall however, this website didn't respond terribly well to a simulated load of 25 users. Over 20% of all requests failed in some way (further investigation with web server logs might be required to establish why). Response times were relatively high and there were significant variations in response times over the course of the test. On today's internet 25 simultaneous users is not a terribly heavy load. This should be investigated further and recommendations made with a view to reducing the incidence of errors and bringing response times down.

Test case examples

2.3 Finalise test environment and document completed tests according to test plan and test logs and result sheets.

W3C HTML Standards

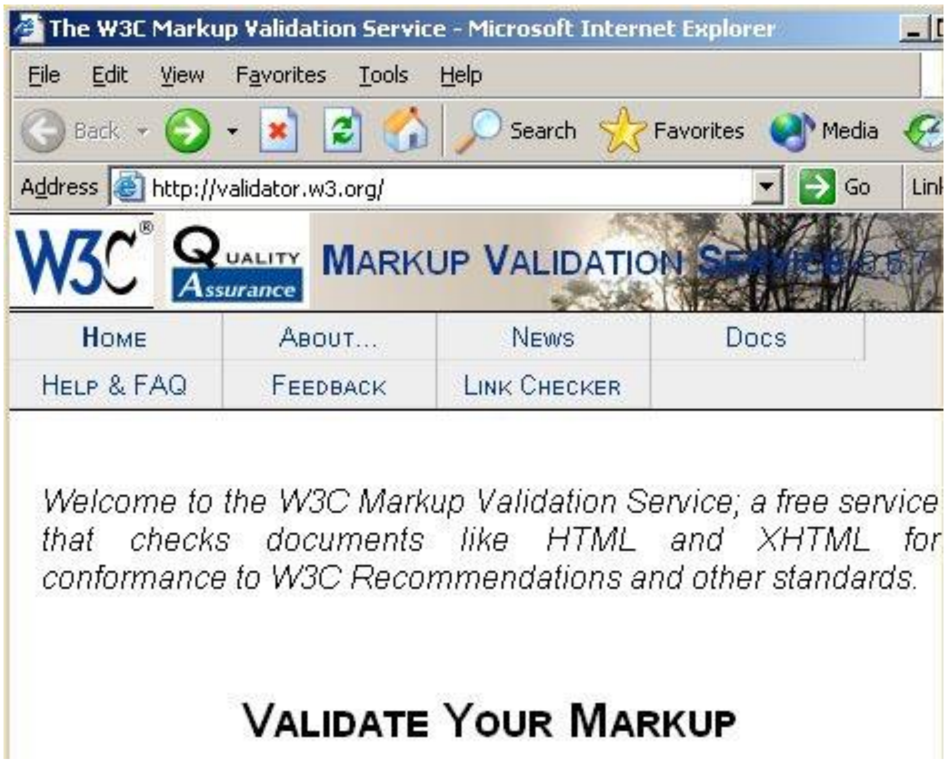
When developing a website it is important to remember that in most cases you have no control over which web browser will be used. Visitors to your site may not be using a "standard" web browser at all (e.g. a person may use a screen reader instead). In the mid-1990's as the web evolved many web browser vendors (most notably Netscape and Microsoft) added their own extensions to the version of HTML that their browser supported. In some cases these extensions were submitted to the W3C (Word Wide Web Consortium) and have since been adopted as part of the formal HTML standards. In some cases, extensions formalised by the W3C may not have been fully implemented by all major browser vendors (e.g. Cascading Style Sheets (CSS) don't always work properly with Internet Explorer).

Many websites on the Internet today only work correctly when displayed in Internet Explorer and don't display properly in other browsers. In some cases this is due to poorly written JavaScript. Where the visible layout is "broken" in another browser the cause is most likely due to poor or "invalid" HTML. The W3C has published a series of documents that set the standards for HTML. They define which tags and attributes are available for use, under what circumstances they can be used and how a web browser should interpret each tag. A web page that uses HTML entirely in accordance with W3C standards is said to contain "valid HTML". A single error however, renders the entire page "invalid".

We should try to ensure that all pages on our websites contain only valid HTML. This ensures that we are not using any proprietary extensions to the HTML standard. Avoiding these extensions gives our web pages a much greater chance of displaying correctly on all popular web browsers available today. Valid HTML can reduce the need for testing our pages on every possible web browser and operating system combination. If we know the HTML is valid, we can reasonably assume that any browser supporting that version of HTML will display the page properly. This is not always the case however and some testing may be required. Using valid HTML does tend to reduce the need for extensive cross-browser testing.

The W3C standards documents are available for download from the W3C website (http://www.w3.org/TR/html401). HTML standards are lengthy documents, the HTML 4.01 standard is 389 pages in PDF form, which describe HTML in intimate detail. Checking web pages for standards compliance involves checking the HTML to make sure it conforms to every recommendation in that 389 page document. This can obviously be a time consuming and error prone exercise if done manually.

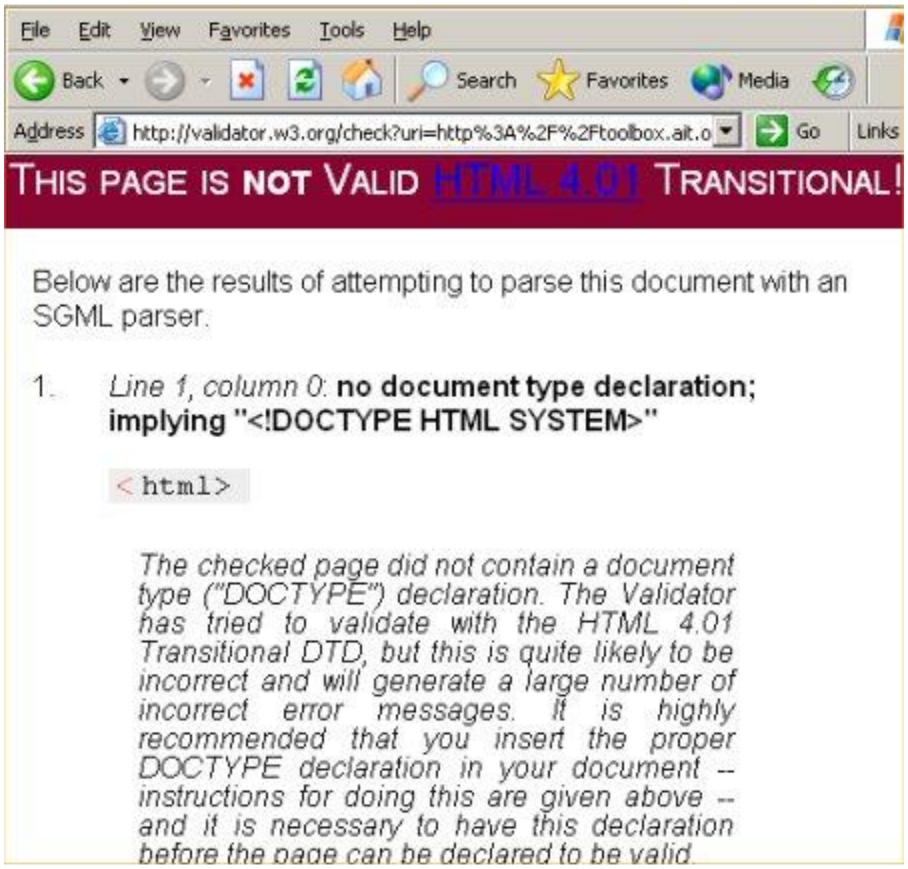
To remedy this problem, several HTML validation programs and services are available. The W3C has their own available online at http://validator.w3.org. which can be used to validate any page on the Internet. The validator is shown in the graphic below:


The W3C Markup Validation Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://validator.w3.org/ Go
W3C Quality Assurance Markup Validation Service
Home About... News Docs
Help & FAQ Feedback Link Checker

Welcome to the W3C Markup Validation Service; a free service that checks documents like HTML and XHTML for conformance to W3C Recommendations and other standards.

Validate Your Markup

The following is a small portion of the validation results for a web page.


File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://validator.w3.org/check?uri=http%3A%2F%2Ftoolbox.ait.o
Go
Links

THIS PAGE IS NOT VALID HTML 4.01 TRANSITIONAL!

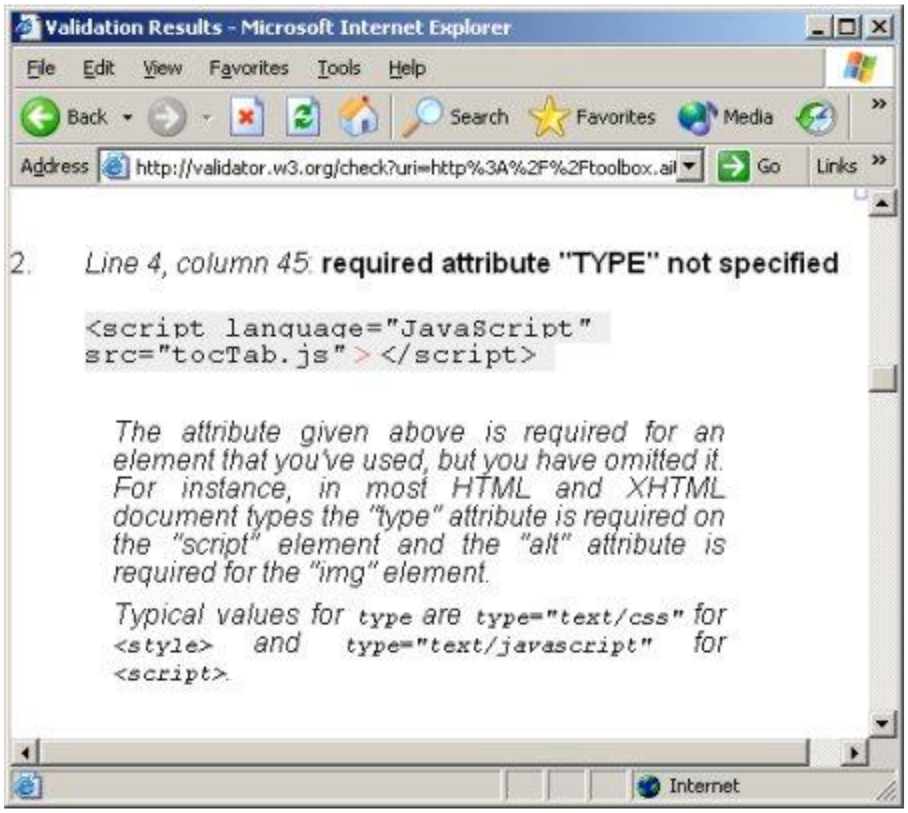
Below are the results of attempting to parse this document with an SGML parser.

1. *Line 1, column 0:* **no document type declaration; implying "<!DOCTYPE HTML SYSTEM>"**

```
<html>
```

The checked page did not contain a document type ("DOCTYPE") declaration. The Validator has tried to validate with the HTML 4.01 Transitional DTD, but this is quite likely to be incorrect and will generate a large number of incorrect error messages. It is highly recommended that you insert the proper DOCTYPE declaration in your document -- instructions for doing this are given above -- and it is necessary to have this declaration before the page can be declared to be valid

First and foremost, this tells us that the page is not valid HTML. Then the individual problems are listed including a short explanation. The first problem arises because there is no DOCTYPE specified for the page. A DOCTYPE tells the web browser what version of HTML the page uses and therefore what rules to apply when displaying it on the screen.


Validation Results - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://validator.w3.org/check?uri=http%3A%2F%2Ftoolbox.ai
Go Links
Internet

2. *Line 4, column 45:* **required attribute "TYPE" not specified**

```
<script language="JavaScript"
src="tocTab.js"></script>
```

The attribute given above is required for an element that you've used, but you have omitted it. For instance, in most HTML and XHTML document types the "type" attribute is required on the "script" element and the "alt" attribute is required for the "img" element.

Typical values for `type` *are* `type="text/css"` *for* `<style>` *and* `type="text/javascript"` *for* `<script>`.

The second problem is a simple HTML coding error, according to the HTML standard <script> tags must include a type="…" attribute. The page doesn't have this required attribute. This is a minor point and doesn't stop the page from working on most browsers, but it is required for standards compliance. There are many other errors on the page, most of which are minor HTML errors. However, the end result is that the website is serving a page which is not valid HTML and therefore may not always display correctly in all browsers and may not be accessible to people with disabilities.

Accessibility

HTML standards compliance alone may not guarantee that a web page is accessible to people with disabilities, but it is a very good start. Tim Berners-Lee, inventor of the World Wide Web states, "The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect." (The Web Accessibility Initiative).

Accessibility issues are particularly important for large businesses and governments who need to make their services available to as wide a cross section of the community as possible. In the USA, website accessibility is required by law for some organisations and given time this may also be the case here in Australia. Equal Opportunity law in Australia may require that your company Intranet is accessible to all workers.

The Web Accessibility Initiative (http://www.w3.org/WAI) from the W3C aims to set standards and guidelines for web accessibility. Mark Pilgrim's excellent eBook "Dive Into Accessibility" (http://www.diveintoaccessibility.org) provides practical advice on making websites more accessible to people with various disabilities.

Privacy Legislation

In Australia any business information that allows you to determine an individual person's identity is covered by privacy legislation. Under the terms of that legislation you must seek the person's consent before requesting any personal information and you may only use the information for the purpose it was collected and purposes related to it. So for example, your website may collect a person's name and address so that you can send them the items they purchased. Under the Privacy Act, you would not be allowed to use that information in a marketing survey, or sell it to a mailing list company unless you explicitly sought permission to do so when the information was collected.

In Australia, the Privacy Act effectively means that every website needs to publish a privacy statement, which details exactly what information is collected about visitors (including things in the web server logs such as IP addresses which may be used to identify people) and what the organisation will do with that information. It would also be a good idea to ensure that the privacy policy is prominently linked from any page that collects personal information. The Office of the Federal Privacy Commissioner (http://www.privacy.gov.au) publishes easy to read Frequently Asked Questions and Guidelines on their website that explain an organisation's obligations under the Privacy Act. When testing websites check for a privacy policy and make sure the privacy policy is prominent and accurate.

The resource pack on **Privacy Policy** discusses privacy legislation in more detail.

Organisational Standards

Your organisation may have certain standards relating to the usage of its logos, colour schemes (and in some cases exact colours) to be associated with the organisation, to ensure consistency across all of its activities, e.g. letterheads, website, TV commercials, print advertisements. It is important that you be aware of these standards and comply with them. Failure to do so may cause a website design to be rejected purely on visual grounds alone.

Your organisation may also have standards relating to handling of source code (e.g. indenting, variable naming, source control, etc.) and layout of documentation. You may be required to write a test plan in a particular format, or address certain points in a particular order. Recording the results of a test may need to be done on special forms, or follow a standard.

Every organisation will have some sort of accepted standard for documentation. It's your responsibility to familiarise yourself with your organisation's standards and ensure they are followed.

2.4 Compare and document actual results to expected results for each system unit and complete result sheets

Throughout testing it is important to keep detailed notes about what tests were performed and the results obtained from each test. When conducting usability testing it may also help to record a video of the test sessions for later review. Testers should also keep detailed notes about what was tested and their results. It's important that all test results be recorded as they occur rather than relying on a tester to remember what worked and what didn't. This is of particular importance during usability testing, as usability tests can't easily be repeated. Repeating a usability test with the same subject will give different results because they'll have learnt from their mistakes during the last test and finding a new subject may not yield the same results.

Some organisations might require the use of a special form to record results during testing, others might require a particular layout be used to document test results. If your organisation has these procedures or policies in place it's important that they are followed during testing. When testing is complete you will want a full and detailed record of the results so that you can evaluate the success of the testing performed.

Once the results of testing have been documented it is important to then reconcile with the original test plan to ensure that all tests were performed as documented and to assess the results of testing against expectations.

Once we're satisfied that all of the tests were performed correctly, we need to compare the results against what was expected. Some tests in the test plan may be designed to fail. For example "Order a product from the website, give credit card number '1234 5432 3456 7654'" should fail because the credit card number is invalid. If the website accepted the order, it should be considered broken because it accepts invalid credit card numbers. If the results obtained agree with the test plan, then the test should be considered a success.

Once testing is complete, a report should be prepared describing the results of the testing. The report should pay particular attention to the tests that did not work as expected as these represent issues that may need to be followed up. The report should also highlight the results of any benchmark tests and tests against acceptance criteria. These items will be required by management in order to get the customer to sign off on the project upon completion.

2.5 Summary

Traditional style module and unit tests also work with websites, however the Internet exposes applications to a whole new audience and introduces its own peculiar set of issues. Load testing may not be critical for a traditional client/server application because the number of users is typically known in advance and does not fluctuate rapidly. In most cases, a traditional client/server application is used by the staff in a single organisation and staff numbers tend to remain relatively constant. However, on the Internet, the potential audience for your website is growing constantly as more people connect themselves to the Internet. Outside forces and links from external sites may drive a surge of visitors to your site without warning (the so called "slashdot effect"). Thus, specialised testing techniques are required for websites, which may not apply to traditional software development.

In this resource pack we've considered some testing tools and techniques in more detail, including some tools specifically for HTML validation, link checking and load testing. You should now have a good understanding of the techniques involved unique to website testing. We've also considered the issues introduced by the federal Privacy Act in Australia and looked at the issues that may create when testing websites.

Further Reading

LinkChecker	http://linkchecker.sourceforge.net
Wayback Machine	http://www.archive.org/web/web.php
Web based ApacheBench	http://codeflux.com/ab
W3C standards documents	http://www.w3.org/TR/html401
W3C Validator	http://validator.w3.org
Web Accessibility Initiative	http://www.w3.org/WAI
Dive Into Accessibility	http://www.diveintoaccessibility.org
Office of the Federal Privacy Commissioner	http://www.privacy.gov.au
First rule of usability? Don't listen to users	http://www.useit.com/alertbox/20010805.html Nielson, Jakob, 2001, Retrieved: 6 July 2004 from
The Web Accessibility Initiative	http://www.w3.org/WAI Brewer, Judy. 2004, Retrieved: 7 July 2004 from
Dive Into Accessibility	http://diveintoaccessibility.org Pilgrim, Mark. 2002. Retrieved: 7 July 2004 from