ICTSAS519 Perform systems tests – Part 1

Contents

1. Prepare for Test

For a business or other organisation, its website can be a highly visible advertisement for the company and its products or services. In most cases if a website doesn't leave a good impression with the end user, the user will move to a competitor's website without ever contacting your company. You might have lost a potentially valuable customer, without even knowing why they chose not to deal with you. When developing and deploying a website, it's worth remembering that the results of your work may well be the first (and possibly only) point of contact with a potential customer and first impressions count!

Broken links, slow pages, bad visual design and layout, or pages which contain error messages, are problems that can leave an overwhelmingly negative impression of a website and the organisation operating it. With a good test plan we seek to minimise the risk and avoid some of these problems altogether.

For e-commerce sites, the bad press resulting from security problems could be devastating. Security testing is essential for any site dealing with sensitive data (such as credit card details), to prevent hackers from gaining access to this data.

In this module we'll look at how to go about testing a website and how we plan our testing process to ensure that the testing provides useful data for all involved.

The topics contained within this resource are:

- Prepare Test Environment
- Determine software life cycle
- Define test plan and appropriate test tool
- Separate Systems into Modules
- Gather and Prepare Logs and result sheets
- Notify Operations of Test
- Prepare Test Scripts and Batches
- Review Expected Results
- Summary

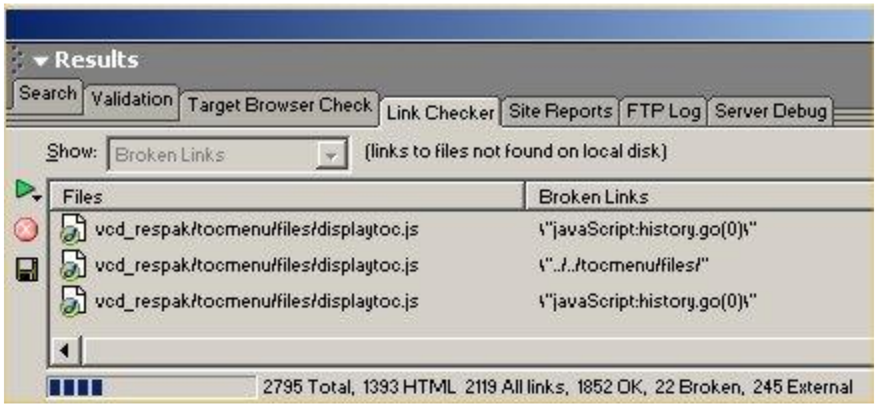
1.1 Determine acceptance criteria from system specifications

There are many different forms of testing which can be applied to websites and each will require its own specialised test environment. For example, if you are load testing a website, you would look to mimic the exact server configuration in the live system, then set up a high speed network with a number of machines running specialised client software. Security testing on the other hand requires quite a different test configuration. Let's have a closer look at some of the different types of test we can conduct on a website and the test environment required for each.

Proofing Content

One of the most important things we can do is check our websites for broken links and proof reading the content for spelling and grammatical mistakes. Broken links are hyperlinks on a web page that point to a location that doesn't exist. Common causes for this include invalid or misspelt domain name or page location in the hyperlink. If the hyperlink points to a third party site, the link may be broken if the third party site is redesigned, moved or taken offline and the content no longer available in its original location. If your website links to third party websites, it's important to check your site for broken links regularly, otherwise links on your site could become broken without you realising what has happened. If you're setting up a new website, it is important to check for broken links after uploading the site to your server to ensure that all pages and images have been uploaded successfully.

Checking for broken links can be done manually, by going through every page and clicking on every link, making sure all pages load correctly and all images are displayed as they should. However, this is tedious and error prone and can easily be automated. Link checking tools may be built into the web design tools you used to design the website (such as Microsoft FrontPage, or Macromedia's Dreamweaver). The graphic below shows the link checker in Macromedia's Dreamweaver:



Show: Broken Links (links to files not found on local disk)

Files	Broken Links
vcd_respak/tocmenu/files/displaytoc.js	\"javaScript:history.go(0)\"
vcd_respak/tocmenu/files/displaytoc.js	\"../../tocmenu/files/\"
vcd_respak/tocmenu/files/displaytoc.js	\"javaScript:history.go(0)\"

2795 Total, 1393 HTML 2119 All links, 1852 OK, 22 Broken, 245 External

You can also get standalone link checking programs for most operating systems. Standalone link checkers can be used to test the internal links in your website even if it is not accessible from the Internet. Finally online testing tools are available through various websites. In order to use online tests, your test website needs to be accessible from the Internet. To use the online tool, visit the website and enter the URL for the site you wish to test. You will then receive a report about your website.

Proofreading for spelling mistakes and grammatical errors can also be done manually, but automated tools also exist. Again, these may be built into your web design tools, but standalone and online tools are also available.

Standards Compliance

When we check a web page or website for standards compliance, we need to check that the HTML, CSS, JavaScript, etc. conforms to the standards published by the W3C (Word Wide Web Consortium). W3C standards determine how HTML tags are used to define and lay out a web page. However, some web browsers implement extensions to the formal standards, which are not available on other web browsers. If your website uses these extensions, then it may not be compatible with some of the many different web browsers commonly in use. Testing a website for compliance with formal web standards is a good step towards making sure that it displays correctly in all web browsers and can also make your web page accessible to people with disabilities. For example, many screen-reading programs don't work correctly if a web page is not valid HTML. Checking for standards compliance is difficult to do manually as it requires an intimate knowledge of the W3C standards for HTML. However the W3C itself has released a tool that will analyse the HTML in your pages and report on any standards related issues. Third party tools are also available.

When testing for broken links, spelling and grammatical problems and standards compliance, your test environment will typically consist of a complete copy of the website running on a web server. The test environment will also need a client machine to run your chosen testing tools.

Usability Testing

The forms of testing mentioned above all relate to the usability of a website, particularly making sure that it is accessible to most web browsers and people with disabilities. On large and complex websites it can also be useful to test the usability of the website's navigation.

- Can people find the content they are looking for?
- Can people figure out how to place an order through your e-commerce site?
- Does the site load pages unacceptably slowly on a dialup connection?

Usability testing can be done in many different ways, from using printouts of the website in an interview, to asking people to try and do various things with the website and observing what they do. The test environment required for usability testing will depend largely on how you plan to conduct the testing.

Load Testing

Most of the Internet is beyond our control. We have no influence over what is written on third party websites and no way of controlling who links to our own websites. In the case of some websites, their primary purpose is to link to other sites which their readers might find interesting (the geek news site Slashdot is a prime example). When these sites gain enough readers, a link from their website to yours can send an enormous amount of people your way in a short period of time.

Sites which have links from Slashdot regularly crash because they simply cannot cope with the sheer number of visitors they receive within minutes of a link being published on the Slashdot home page. This has become known as the "Slashdot effect". Similarly, events in the real world may drive people to certain websites, e.g a TV advertising campaign with your website address may bring an unusually high number of visitors. However it may happen, we can't control (and may have no way of knowing in advance) how many visitors our websites may receive. This makes load testing particularly important, because through load testing we can establish how many simultaneous visitors our website and web server can deal with. Load testing may also allow you to formulate a plan for dealing with high traffic periods and allows you to test such plans before they need to be implemented.

When load testing a website, it is critical that you replicate as closely as possible your production web server's hardware and software configuration. Factors such as processor speed, amount of RAM and disk/network speed can play a big part in determining how many visitors a site can handle. Website content also plays a role here as well; larger images and bigger pages will take more resources to serve than smaller ones.

Dynamically generated pages (e.g. PHP or ASP pages) will take more resources to serve than static HTML. Streaming video and audio also require more resources to serve than pictures and text. Slight variations in hardware specifications, software configuration or website content may lead to large variations in the number of users your site can deal with. So when setting up for load testing, you'll need to try and replicate the live production environment in order to get meaningful results.

The most effective way to load test a website is to use a specialised testing tool specifically designed for this purpose. There are many on the market, including ApacheBench (a benchmarking tool for Apache). Load testing tools such as ApacheBench allow you to specify how many times to request a particular page and how many simultaneous requests to make. It then measures the time taken for pages to be returned and reports the results. In order to accurately load test your website you may need multiple client machines running load testing tools to simulate a high level of traffic.

Security Testing

Security testing is critical for most sites, for many businesses (particularly e-commerce sites) the bad publicity resulting from a compromise of sensitive data would be disastrous. There are two different approaches to security testing, both of which have a role to play in making sure your website is secure.

The most effective security test for a dynamic website is to take the source code and give it to someone who hasn't worked on that code before and ask them to review the code for security defects. In a large team, you might swap modules with a co-worker and review each other's work. Smaller teams might contract an outside developer for their code reviews. The Open Web Application Security Project (OWASP) publishes a list of what it considers the top 10 security issues with web applications. Every web developer should be aware of these issues because the security problems inherent in web applications are quite different to those in traditional GUI programming. Code reviews for security can be performed in the normal development environment and don't require special testing configuration.

Another method commonly used for security testing involves setting up a replica of the production server configuration and trying to break into it. This is known as "penetration testing" and can be used to find bugs that may have been missed in code reviews. The person conducting the penetration test will need to have an extensive knowledge of website security techniques and a good understanding of how web applications are commonly "hacked". To increase the chance of the penetration tester finding problems, you may also wish to allow the penetration tester access to the site's source code (something a normal hacker may not have).

Test Environments

As we've seen, there are many different types of tests we can perform on our websites and the test environment for each can be quite different. In most cases though we will want to test a copy of the website on a separate system. In this way we avoid disturbing developers or production systems in order to conduct our testing. For some forms of testing (e.g. load testing) it may be worth setting up an isolated test system to avoid causing problems for other users of our networks. In most cases you'll want to test the website in an environment that is as close to your "live" or production environment as possible. This may mean using the same (or very similar) server hardware, using "live" data in databases and simulating normal (and abnormal) levels of traffic. It may also mean setting up various dialup and broadband internet connections for accurate speed testing. If your website uses commercial software (e.g. Operating System, Web Server software, etc.) is it also important to ensure that all of the software used in the test environment is properly licensed. Even if the test environment is temporary, you will still need to ensure that adequate licences have been purchased to allow the test lab to operate legally.

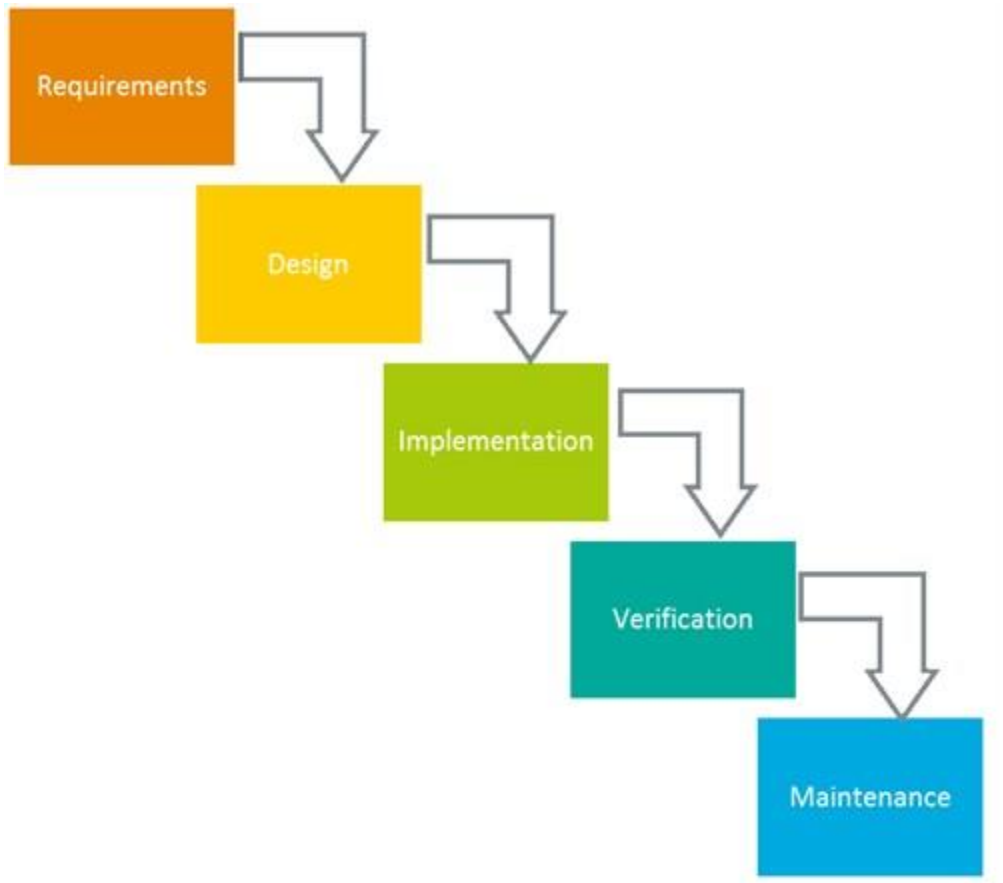
1.2 Determine and document software life cycle according to system specifications and contact in operations

Selecting an Appropriate Software Development Lifecycle (SDL) Model in an Agency Environment

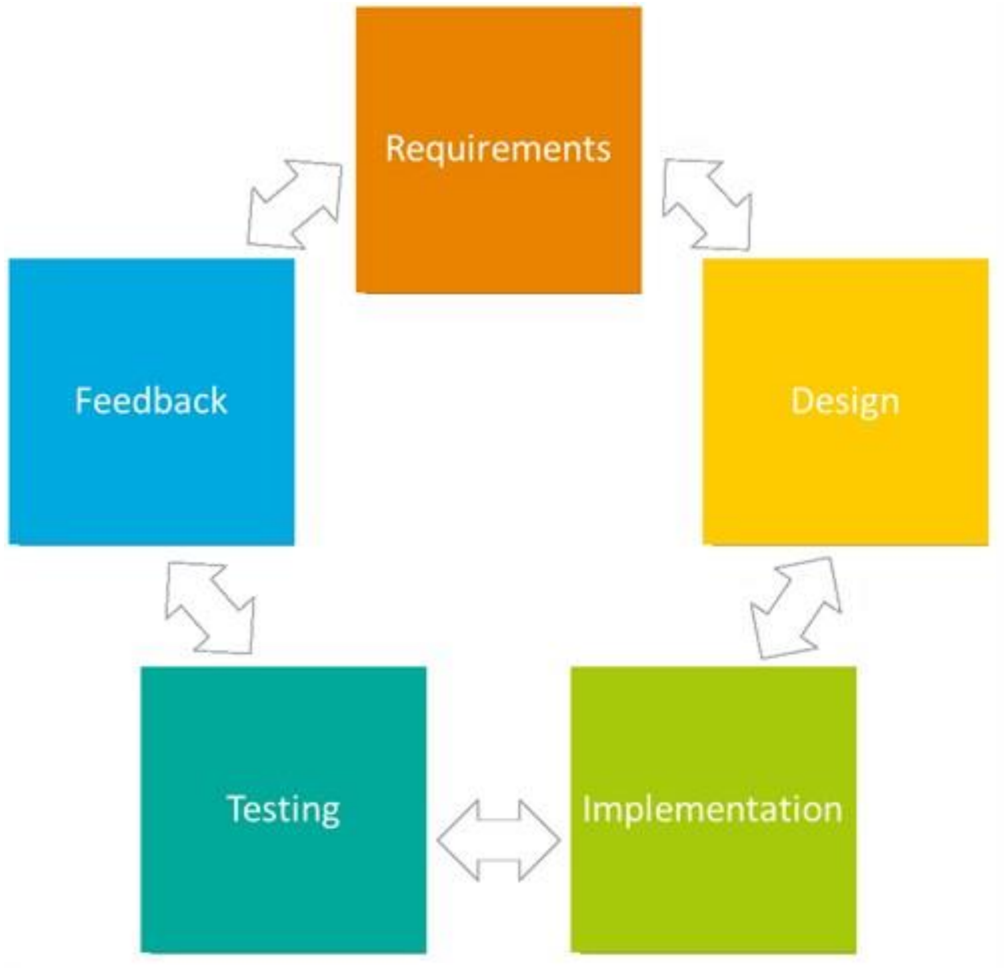
The choice of a relevant and effective development model is one of the key factors in determining the success or failure of a software company. Key factors that influence the decision are the type of product being created, time restraints, and the resources available to support the chosen model.

By far the two most common models in use today are the Waterfall Model and Agile Development. The various benefits and drawbacks of these two models have been discussed in detail elsewhere, so I'll simply offer a quick summary.

The Waterfall Model involves planning the entire development of a project from the initial idea to its completion in a step-by-step sequential process. Each phase cannot begin before the previous phase is completed. The model is best used in smaller projects where the product is extremely well understood from the outset, with well-laid-out timelines.



Agile Development is based on the idea of completing bits of the work in small chunks called sprints. The product is built bit by bit, feature by feature, and at each stage, analysis takes place to see if the product needs to evolve from its original scope. This process works best on larger projects that are easily divided into discrete features.



However, in an agency environment, where the client and the producer of software are two different entities—not always with the same goals and opinions—which development model is going to work better? The answer is: neither and both.

Agency projects are usually generated by clients' ideas. These ideas are often loose and ambiguous, and it's the agency's job to translate them into something that will suit the client's needs. Thus, the best strategy when beginning a project is to treat it like the Waterfall Model. Design the product and timeline in whole and get client sign-off on the plan.

However, in the fast-paced churn of agency development and project-based work, it is a mistake to expect that the original plan and design will not change. Because each project is different, and there is limited time for discovery, you rarely know what you are getting into during the planning phase. A client personnel change during the project, technical limitations discovered during development, design implementations not working out—there are a hundred reasons why changes might need to be made on the fly during the project. The most common breakdown of the Waterfall Model is during the QA phase, where many of these problems come to light.

So how do you get over this hurdle? Our approach is simple—by planning the Waterfall Model through QA as only the first part of development. After the QA phase, abandon the original plan and establish a new timeline to work in a more agile way. Create sprints based on the remaining work and the client feedback, keeping the client in the loop regarding how much time and resources each sprint will consume. This process lets the client see changes rapidly and assures them that they are being listened to and that the agency is devoted to producing the product that will actually fit their needs, not just sticking to a flawed plan.

Which model works better for your project? It all depends on the project's size and complexity. Smaller, simpler projects usually benefit from the Waterfall Model, while larger, more complex projects are a better fit for the agile model.

The key is to be flexible and find what works for your agency and your clients.

1.3 Define test plan from acceptance criteria, software life cycle, system specifications and in compliance with organisational testing and acceptance processes

A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. A test plan is usually prepared by or with significant input from test engineers.

Depending on the product and the responsibility of the organization to which the test plan applies, a test plan may include a strategy for one or more of the following:

- *Design Verification or Compliance test* - to be performed during the development or approval stages of the product, typically on a small sample of units.
- *Manufacturing or Production test* - to be performed during preparation or assembly of the product in an ongoing manner for purposes of performance verification and quality control.
- *Acceptance or Commissioning test* - to be performed at the time of delivery or installation of the product.
- *Service and Repair test* - to be performed as required over the service life of the product.
- *Regression test* - to be performed on an existing operational product, to verify that existing functionality didn't get broken when other aspects of the environment are changed (e.g., upgrading the platform on which an existing application runs).

A complex system may have a high level test plan to address the overall requirements and supporting test plans to address the design details of subsystems and components.

Test plan document formats can be as varied as the products and organizations to which they apply. There are three major elements that should be described in the test plan: Test Coverage, Test Methods, and Test Responsibilities.

Test methods in the test plan state how test coverage will be implemented. Test methods may be determined by standards, regulatory agencies, or contractual agreement, or may have to be created new. Test methods also specify test equipment to be used in the performance of the tests and establish pass/fail criteria. Test methods used to verify hardware design requirements can range from very simple steps, such as visual inspection, to elaborate test procedures that are documented separately.

A web application performance tool (WAPT) is used to test web applications and web related interfaces. These tools are used for performance, load and stress testing of web applications, web sites, web servers and other web interfaces. WAPT tends to simulate virtual users which will repeat either recorded URLs or specified URL and allows the users to specify number of times or iterations that the virtual users will have to repeat the recorded URLs. By doing so, the tool is useful to check for bottleneck and performance leakage in the website or web application being tested.

A WAPT faces various challenges during testing and should be able to conduct tests for:

- Browser compatibility
- Operating System compatibility
- Windows application compatibility where required

WAPT allows a user to specify how virtual users are involved in the testing environment.ie either increasing users or constant users or periodic users load. Increasing user load, step by step is called RAMP where virtual users are increased from 0 to hundreds. Constant user load maintains specified user load at all time. Periodic user load tends to increase and decrease the user load from time to time.

Web testing tools may be classified based on different prerequisites that a user may require to test web applications mainly scripting requirements, GUI functionalities and browser compatibility.

Some examples of automated test tools

Test Plan Template

1.4 Notify contact in operations of scheduled tests to understand implications for operations and modify schedule to minimise implications for operations

As websites grow increasingly larger and more complex, testing the entire site becomes difficult. A relatively simple news or product information site could have hundreds of pages that may need testing. An e-commerce site retailing only a few hundred products may contain a thousand or more pages of product information, all of which may need to be tested. When testing large websites, it becomes very important to be able to recognise the different features or modules that make up the site and test them individually.

Identifying Different Users

Different people may use a website in different ways depending on their interest in the site at that time. Consider the popular auction site eBay (http://www.ebay.com.au) as an example.

Example

Lets say you're a first time eBay user. You might start by browsing the site looking for something that interests you. Once you find an item, you'd need to register for a user account and place a bid. A more experienced eBay user might visit the site, log in to their user account, check the items they're bidding on, run a search or two and leave.

Example
An eBay seller might log into their account and check on the items they're selling, or list a new item for sale. Once bidding has completed on an item, both buyers and sellers need to log in and complete the transaction. An eBay administrator might log in and add a new category for listings or the eBay complaints department might need to remove an item from sale after a complaint.

Without delving too deeply into the functionality of this complex website, we can identify at least six different ways in which people can use the eBay website. When planning to test a website, it helps to identify as many of these different user profiles as possible and many common tasks as possible. In that way we can divide the testing up into smaller tasks that better represent how people actually use the system. This is an important concept for usability testing. If you are able to identify a particular group of users, you could then bring in representatives of that group to test your new website. If you carefully select your test subjects and ask them to test relevant features, you will improve the quality of the feedback you receive.

By identifying the different users who might use your system you can then plan different tests to ensure that features important to each different type of user are fully tested. Testing realistic user scenarios also means that we can better test the interaction between the different modules of the system to ensure that they work together in ways that the user is likely to use.

Identifying Different Modules

On an e-commerce website, you might have a search function which allows you to search for a specific product and a catalogue function allowing you to see what is available and pick an item. Let's say we do two tests:

- Firstly use the browse function to find a product, click its details page and add it to the shopping cart. We can then check the cart and make sure the item is there.

- In the next test, you search for a product, select the desired product and go to its details page.

It might be easy to make the assumption that the details page we see now and the details page you get with the browse function, are the same and stop the test there. By identifying both methods of performing a task and testing them from start to finish as a normal user, we may find a bug which would otherwise go undetected.

Once we've identified the different types of users that may use our website and the different functional modules, we can then plan what sort of testing we'll do on each area. For example, usability testing might be more important for the public side of an e-commerce site, than the administrative interface. After all, website administrators may receive training for their roles, but members of the general public need to be able to buy from the site without assistance. Security testing is probably going to be more important for the payment function of an e-commerce site than it is for the help system. By breaking up our system into smaller modules and focusing on how each user will use the site, we can focus our testing on the areas where it will deliver the best results.

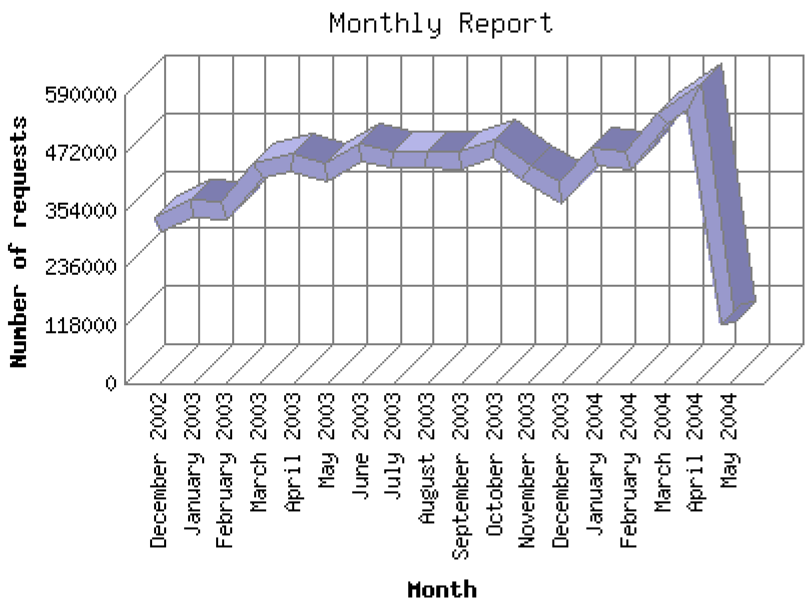
1.5 Develop test scripts for online test and test run for batch test according to test plan

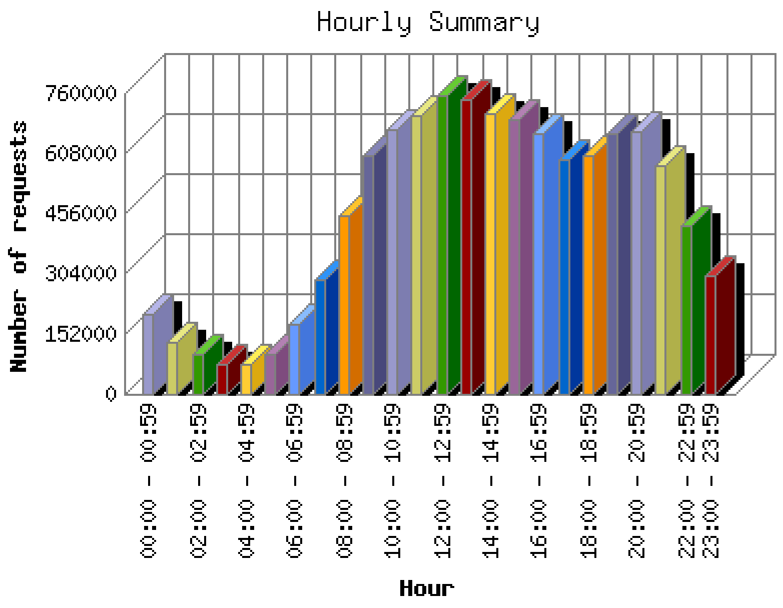
If your website has been in active use for a while, the log kept by your web server can be an invaluable tool when planning tests. Your web server logs will contain details of every page served and usually information such as:

- The date and time
- The number of bytes served
- The status (successful or error number)
- The referring page
- Any query string that was part of the URL

If you're planning a load test, your web server log can give you invaluable information about the number of hits your site is receiving and the times of day which receive the most traffic. The following two graphs were generated from the logs of a reasonably well-trafficked website belonging to a nation wide Australian company.

The first graph shows the total hits per month and clearly shows the growth of traffic to the site over an 18 month period. The second graph analyses the number of hits per hour and clearly demonstrates that website traffic varies considerably across a 24 hour period.





Using these graphs, we can clearly see how much traffic the website currently receives and draw some conclusions about the amount of traffic the website will receive in the future. If this sort of data is available, it gives us some sensible benchmarks to aim for when load testing the site. If we were creating a new website for the company, we know what sort of load the site will need to endure over the course of a month and can test appropriately.

Log files also record which pages return errors, which may give us an indication as to where we might need to focus our testing. If a particular URL returns a 404 "File Not Found" error on a regular basis, then you may have a broken link within your site. Planning to test your site for broken links might be a good idea. If your log records the referring page you might be able to contact the owner of the external site linking to you and ask them to update their link, or set up a redirection on your server so the URL returns some useful information instead of an error.

By analysing your log files you can also determine which pages on your website receive the most traffic. This allows you to focus your testing on the most visible areas of your site.

There are many log file analysis tools available and their cost can vary quite dramatically, from free and open source software, up to multi-thousand dollar commercial packages or services. Essentially though, all of these do the same job. They turn the mass of raw data in the web server logs into more usable information, such as the graphs you see above. When planning website testing, check the data available from the log files of the existing website (if any).

1.6 Prepare test environment and select test tools according to test plan

Unless you have a completely isolated test environment, with dedicated equipment that is not connected to any other networks, your testing may have some impact on other users. If you're sharing equipment with other users, then it is good practice to notify the network administrators and/or network operations of your planned testing. This is especially important if you are planning load testing or security penetration testing.

When conducting load testing the typical objective is to find out how much traffic your website can handle before performance degrades to an unacceptable level. In many cases the limit may not be as a result of the website coding, but hardware factors. Your web server may run out of memory, or use excessive amounts of CPU time during testing. Database access may become a bottleneck rendering the database server unbearably slow. If the server itself can keep up with the load, your network may instead become flooded with traffic, preventing other users from accessing the server.

In many smaller organisations, there may not be sufficient budget or equipment to set up a dedicated test environment. This means you'll need to carefully plan your testing to minimise the impact on other users. This may include testing outside of normal business hours. If you are performing testing on production servers, log file analysis tools may help you identify the most suitable (quietest) times for testing.

When conducting security penetration testing, it is imperative that the network administrators are aware of this, so they are forewarned when strange "hackerish" things show up in the log files (e.g. invalid login attempts). The network administrator should also be given the authority to call off penetration testing if they believe the tester is likely to damage the server or its configuration in any way. Network administrators may also have Network Intrusion Detection Systems in place which will be triggered if an attack on a website is underway. Administrators need to be aware that suspicious activity is to be expected so they can either adjust or disable their detection systems.

Every organisation will have a different procedure for notifying the network administrators. You may simply need to send them an email, or log a request in a helpdesk database. In other instances the request might need to be made in writing, or tabled at a weekly department meeting. The procedure for contacting the network administrators should be documented in a policy document and this should be followed. Your testing can have an impact on the other users of the network or server, so it is important to liaise with the relevant people.

1.7 Prepare test logs and result sheets according to test plan

In the previous sections we've considered what types of testing we might need to perform and the type of test environment we may need to set up. We've examined our website to determine which modules we need to test and the different ways in which people might use the site. We also considered the importance of reviewing the log files from the live website for common errors and high traffic spots.

Risk Management

Database driven websites involve a complex interaction between web browser, web server, operating system, CGI or scripting features, databases and firewalls. This complexity makes it virtually impossible to test all scenarios. Web projects are typically on an aggressive schedule, so some form of risk analysis may be necessary to determine where to focus testing efforts. Consider how closely the test environment will mimic the live production setup and whether the differences will be significant. They may not be for usability testing, but significant differences may make load testing irrelevant. Will the test scenarios closely mimic real-life users, Internet connections, hardware, loads, database table sizes and so on? Using smaller database sizes, or faster internet connections may improve results when testing for download speeds or load testing, but may not accurately reflect the performance of the site when it goes live.

Features that are critical to the site's purpose (eg ordering on an e-commerce site), or those likely to cause complaints or bad publicity are excellent candidates for usability testing. Areas with the heaviest database access, the most complex scripting, or the most popular, should be load tested to ensure they can sustain the number of visitors expected.

A good approach seeks to test as much of the website as possible, with tests that are most appropriate for each module or function. We should spend proportionally more time testing areas with a greater risk of failure (eg critical functions, high traffic pages or those likely to cause complaints) and spend less time on things less likely to cause problems (eg the "Corporate profile" page).

Security Testing

For security testing you'll need to document which modules are to be tested. Your risk assessment will most likely suggest that the modules dealing with sensitive information (such as personal data, or credit card numbers), or displaying input from the end user are the ones most likely to require security testing. You should also consider whether security will be tested via code reviews or some form of penetration testing. In either case, you should document which modules are to be tested and what specific features you want the tester to focus on. If you want your tester to focus on a particular class of vulnerability (such as SQL Injection or Cross Site Scripting) you will need to document this.

Load Testing

Once you've decided which modules need load testing, you should document your choices and set some sort of target (e.g. less than 200ms response time with 100 concurrent users). The target might be set for you as part of the requirements specification and will vary considerably from one site to the next. For example Yahoo! demand that their systems spend no more than 100ms (0.1 seconds) of processing time generating any page (Rasmus Lerdorf in a presentation given at Linux.Conf.Au 2004).

Load testing is usually performed with the assistance of automated software. So once you've decided what to load test, you will need to create the required test batches for your software. Most load testing packages have a function that allows you to use a web browser to record a session with the website, which can later be replayed.

When you actually conduct the load testing, the testing software will replay each recorded session repeatedly to simulate multiple users. Most test systems can replay several test scripts at once to simulate several simultaneous users, while measuring the response times for each user.

Load testing software measures both the response time and the number of errors returned by the website under test. The former feature can be used to measure page loading times on slow (eg dial-up) internet connections.

1.8 Conduct walk-through with superior to review expected results against acceptance criteria and incorporate feedback

For most projects there are a series of requirements, which are absolutely critical to the success of the project. These requirements are usually outlined in the project's functional specification. If we can't demonstrate that the project we're trying to deliver meets the client's acceptance requirements, then they may not be prepared to sign off on the project. We should review the test plan created earlier against the acceptance criteria.

Certain aspects of the acceptance criteria may require specific types of testing. For example one of the acceptance criteria may be:

> "The website should respond to any request in under 500ms with 100 concurrent users".

This effectively requires that you perform a load test on the completed website to demonstrate that the website meets this criterion.

Your specification might also demand that the website is usable in Internet Explorer, Netscape, Mozilla, Mozilla Firefox, Safari and Konquerer browsers and be accessible to people with disabilities. In that case, your test plan must reflect this. Unless you actually test the site with these browsers you can't demonstrate that you've met the acceptance criteria.

1.9 Summary

When testing a website, there are many factors to consider. Modern dynamic websites are extremely complex, making it impossible to test every possibility. It is important to weigh up the risks inherent in each part of a website and focus your testing efforts on the highest risk functions of the site. We've looked at identifying different types of users and breaking the test up into smaller modules. We've seen how we can use web server logs to identify which modules might need particular attention when testing. Finally we've looked at using the test plan to demonstrate that the website meets a customer's acceptance criteria.

Further Reading

eBay	http://www.ebay.com.au
Webaliser	http://www.mrunix.net/webalizer
MrUnix.net	http://www.mrunix.net
ApacheBench	http://codeflux.com/ab
Open Web Application Security Project Top 10	http://www.owasp.org/documentation/topten Retrieved: July 4th, 2004
PHP Tips and Tricks	http://talks.php.net/show/lca04 Lerdorf, R 2004, Retrieved: July 6th 2004