Testing strategy for internet applications

1. Introduction

With the emergence of e-commerce sites and the growth of business performed over the web, it is critical for companies to have their internet (web) applications tested extensively to ensure that they function correctly, are compatible over many different browsers and operating system configurations and can handle a large amount of concurrent users. However, the unique characteristics of web application testing and lacking of adequate tools make the need to develop strategies, methodologies and tools for test of web applications essential.

Internet applications are software programs or applications that receive input and deliver output through the web, usually in the form of HTML or XML. Web applications are dynamic, interactive, often serve as the front end of complicated applications which often involve database at the back-end. With the rapid development of web technologies such as CGI, JSP, PHP, ASP. NET and Microsoft. NET, internet applications are become more and more complicated.

The repercussions of having a poorly operating website are staggering, and even affect the brick and mortar stores that the websites are enabling online. A recent study showed that when errors are found on an e-commerce website, 28% of the people stopped shopping at the site, 23% stopped buying from the site, and 6% of the people were so upset, that they stopped buying at brick and mortar store that the site is based on (Gerrard, 2000a). One can only surmise that the customers feel that if the company cannot

provide a quality website, then they may not be able to sell a quality product from their stores.

Software testing has been studied for years by researchers. A number of testing techniques like Black-Box testing, White Box testing, equivalence partitioning and structural testing have been developed for software testing. Although web applications testing has much in common with the testing of most other client/server applications, the nature of web application pose unique software testing challenges.

The methods to test the e-commerce application are basically similar to those for testing conventional systems. Code-based testing can be used for unit testing, and Specification-based testing for system testing.

Integration testing of e-commerce applications is complicated because a typical application usually has multiple layers of software. Each layer may be written by a different language, and may be running a different protocol. Also, unlike mainframe and client-server applications, e-commerce applications depend on numerous components that must function together, although they might not have been deployed with each other in mind. In any case, all business logic must be thoroughly tested.

Web applications developers and website quality assurance managers need tools and methods that meet their specific needs to test web applications. However, lacking of adequate tools as well as limited capabilities of current tools are making automatic web applications less popularized than it should be.

This paper will propose a testing strategy for Internet E-Commerce applications and assess its strengths and weaknesses. The testing strategy mainly focuses on the testing of the web components.

2. Types of Web Applications

Developing a strategy for testing a particular web application can vary widely depending on the purpose, audience, scope, delivery media, functionality and the kinds of technology used to deliver the application functionality.

2. 1 Internet Presence

This is a simple web site to provide basic information. Typically, there may be limited dynamic processing, such as the ability to submit information via forms. However, the primary purpose is to provide information about an organization. This is usually the first step in staking a claim on the Web. The risk level of this type of site is typically low, since content correctness is the primary concern and the easiest to verify.

Primary Test Concerns: Correctness, usability, compatibility.

Secondary Test Concerns: Performance, security

2. 2 Internet e-commerce

This is a web site designed to promote business via the Internet. In its simplest form, an e-commerce site may point to a number of ways to receive payments and place orders, but the context of e-commerce normally refers to sites which allow customers to browse products, place orders and obtain

product information. These are the online storefronts that can bring in millions of dollars per day for some companies. The audience is sometimes hard to predict in terms of numbers and traffic patterns, which has a large impact on planning server support and load testing. Since an e-commerce site may be simple or complex, the risks can range from insignificant to extremely high.

Primary Test Concerns: Correctness, security, performance, usability, compatibility, integration

Secondary Test Concerns: Visibility

2. 3 Intranet

This is a web site used internally in an organization to deliver information and functionality. Unlike the Internet sites, the intranet audience is usually well-defined and predictable. Intranets can be used to deliver functionality such as Enterprise Resource Planning, including HR, inventory, accounting, etc. The risks of this type of Web application can be very high in the case of ERP applications.

Primary Test Concerns: Correctness, usability, security, integration

Secondary Test Concerns: Performance

2. 4 Extranet

This is a web site used to communicate with external entities such as customers and suppliers. The audience is well-defined and predictable, but the external nature of the access opens a security risk. Extranets can be

used for business-to-business e-commerce. The risks of this type of Web application can be very high in the case of business-to-business applications.

Primary Test Concerns: Correctness, security, usability, compatibility, performance

Secondary Test Concerns: None

2. 5 Internet media distribution

This is a web site for delivering video and audio over the Internet using technologies such as Real Media and MP3. In this type of site, the correctness of information is less important than the quality of the media delivery. Therefore, performance is a key test concern, as well as the ability to provide quality content on multiple platforms. The risks of this type of Web application can be very high in terms of customer satisfaction and site reputation.

Primary Test Concerns: Performance, usability, compatibility.

Secondary Test Concerns: Correctness

2. 6 Non-Internet media distribution

This is a web application delivered via CD-ROM that uses web technology to deliver information such as training, catalogs, etc. These applications typically are more concerned with delivering quality media content as opposed to functional correctness. Functional correctness is normally limited to searches, navigation and information. The audience is a single user as opposed to thousands of concurrent users. Compatibility on multiple

browsers is a test concern. The risks of this type of Web application can be very high in terms of company reputation.

Primary Test Concerns: Usability, compatibility, correctness.

Secondary Test Concerns: Performance, security

2. 7 Non-Internet application

This is an application using browser functionality and delivered via CD-ROM, but not delivered over the Internet. These applications can be very complex, depending on the application. Functional correctness is a major test concern, as well as compatibility on multiple browsers. The risks of this type of Web application can range from low for simple applications to very high in the case of complex applications.

Primary Test Concerns: Correctness, usability, compatibility.

Secondary Test Concerns: Performance

2. 8 Focus of this paper

The starting point for developing a Web testing strategy is to first identify the type of application you will be testing. This will allow you to eliminate the tests that will have little payback and focus on those tests that relate to the high risk factors. In this paper, we will concentrate our discussion on the testing strategy for Internet E-Commerce applications.

3. Differences between Web and Traditional Client-Server Systems

We will begin to explore additional differences between Web and traditional systems so that appropriate considerations can be formulated specifically for testing Web applications.

3. 1 Client-Side Applications

As illustrated in the Figure 3. 1, most client-server systems are data access applications. A client typically enables users, through the user interface, to send input data, receive output data, and interact with the back end. Clients of traditional client-server systems are platform-specific. That is, for each supported client platform (e. g., Windows 16- and 32-bit, Solaris, Linux, Macintosh, etc.), a client application will be developed and tested for that target platform.

Most Web-based systems are also data access applications. The browser-based clients are designed to handle similar activities to those supported by a traditional client. The main difference is that the Web-based client is running in the context of a Web browser. It renders static HyperText Markup Language (HTML) as well as active contents to display Web page information. Several popular browsers also support active content such as client-side scripting, Java applet, ActiveX control, cascading style sheet (CSS), dynamic HTML, security features, and other goodies. In making these software components, incompatibility issues are introduced among various browsers and their releases.

I3. 2 Server-Based Applications

Server-based applications are programs that don’t have a user interface with which the end users of the system interact. Instead, the client application interacts with server-based applications to access functionality and data via communication protocols, application programming interface, and other interfacing standards.

To black-box testers, server-based applications are black boxes. Server-based applications or systems are often isolated away from the end users. When a server-based application fails, as testers or users from the client side, we often don’t know when it failed, what happened before it failed, who was or how many users were on the system at the time it failed, and so on. This makes bug reproducibility even more challenging for us.

In testing Web systems, we need a better way to track what goes on with applications on the server side. One of the techniques used to enhance our failure reproducibility capability is event logging. With event logging, server-based applications can record activities to a file that might not be normally seen by an end user. When an application uses event logging, the recorded information that is saved can be read in a reliable way.

Figure 3. 1 Client-server versus Web-based clients

3. 3 Web Systems

The complexities of the PC model are multiplied exponentially in Web systems (Figure 2). In addition to the testing challenges that are presented by multiple client PCs, the server side of Web systems involves hardware of

varying types and a software mix of OSs, service processes, server packages, and databases

3. 3. 1 Hardware Mix

With Web systems and their mixture of flavors of hardware to support, the environment can become very difficult to control. Web systems have the capacity to use machines of different platforms, such as Unix, Windows NT, and Macintosh boxes. Such hardware mixtures present testing challenges because different computers in the same system may employ different OSs, CPU speeds, buses, I/O interfaces, and more. Each variation can potentially cause problems.

3. 3. 2 Software Mix

At the highest level, as illustrated in Figure 3. 2, Web systems may consist of various operating systems, web servers, application servers, middleware, e-commerce servers, database servers, major enterprise resource planning (ERP) suites, firewalls, and browsers. For Web systems, software is often designed to run on a wide range of hardware and OS combinations, and risks of software incompatibility are always present. Another problem inherent in the simultaneous use of software from multiple vendors is that when each application undergoes a periodic upgrade (client or server side), there is a chance that the upgrades will not be compatible with preexisting software.

Figure 3. 2 Web system architecture

3. 4 Interoperability Issues

Interoperability is the ability of a system or components within a system to interact and work seamlessly with other systems or other components. This is normally achieved by adhering to certain application program interfaces (APIs), communication protocol standards, or to interface-converting technology such as Common Object Request Broker Architecture (CORBA) or Distributed Common Object Model (DCOM). There are many hardware and software interoperability dependencies associated with Web systems. It is possible that information will be lost or misinterpreted in communication between components. It is essential that our test-planning process include study of the system architectural design. Figure 3. 3 shows a simplified Web system that includes three box servers and a client machine.

Figure 3. 3 Interoperability

3. 5 Piggyback Off Existing Systems

We should be using as much of the existing system (non-web-based) as possible. Ideally, most of your server-side processing will be done using existing systems. This way, you can use your existing tests to test much of your server-based functionality

Figure 3. 4 Testing methods

3. 6 Gray Box Testing

Web testing techniques can also be termed as Gray-Box Testing :

* Incorporates elements of both black-box and white-box testing

* Evaluates application design in the context of the interoperability of system components

* Consists of methods and tools derived from the knowledge of the application internals and the environment with which it interacts

* Considers the outcome on the user end, system-specific technical knowledge, and the operating environment

* Is well-suited to testing Web applications because it factors in high-level design, environment, and interoperability conditions.

* Gray-Box testing will reveal problems that are not as easily considered by a black-box or white-box analysis, especially problems of end-to-end information flow and distributed hardware/software system configuration and compatibility

Not every tester in a group needs to be a gray-box tester. A mix of different types of testers will likely provide the best results.

4. Key areas of Focus for web application testing

The key areas of focus are people, tools, process and environment :

4. 1 People

* training in how to test web-based applications

* understanding of the risk associated with your organization’s web applications

* management support of the testing process

* accuracy of project schedules, especially concerning testing

* cooperation between the testing organization and the web developers

4. 2 Tools

* awareness in your organization concerning automated test tools and how they can be used in testing web applications

* tool ownership in your organization

* tool usage in your organization

* management support for the purchase of test tools in your organization

* management support for the use of test tools in your organization

4. 3 Process

* process definition for developing web applications

* process definition for testing web applications

* process definition for configuration management in the web environment

* acceptance for processes in your organization

* standardization for web development in your organization

4. 4 Environment

* test environment dedication for testing only

* test environment control

* tool support in the test environment

* test data management

* test environment mirroring the production web environment

5. Detailed Testing Process

Matrix model of the test process

Goal

Processes

1. Set and agree on realistic expectations for the system

* Prioritize business requirements; identify those that are essential and those that are optional.

* For a given delivery date, filter out, reprioritize, or renegotiate unrealistic system expectations.

* Specify requirements that are testable. Remove or firm up ambiguous requirements.

* Define acceptance criteria that are meaningful to the business

* Define relevant industry standards or competitive benchmarks.

* Ensure that stakeholders agree to a capacity plan for the system with measurable capacity requirements.

2. Define a test strategy

* Formulate and document a test strategy; relate it to the test quality plan.

* Identify high-risk zones early on; mark these for early testing and keep them away from the project’s critical path.

* Identify all aspects of the system that require testing; in each case, indicate the level and form of testing.

* Identify relevant testing tools that can help automate or document elements of the test process.

3. Plan the testing

* Estimate required resources, such as tools, people, hardware and software infrastructure, test location, and so on.

* Define roles and responsibilities; identify a suitably qualified test manager and test team.

* Draw up and agree to a detailed test plan with the client; define test milestones and test deliverables.

* Plan a pre-release simulation period where the system can be observed under simulated business usage.

4. Set up the test environment

* Set up the physical test environment-for example, the hardware and software infrastructure.

* Prepare test material such as detailed test plans, test scripts, and test data. Prepare test materials with reuse in mind.

* Set up the defect-tracking system; decide on the form of defect reporting and any test metrics to use.

* Define test standards and procedures; educate individuals on the use of standards and procedures.

* Acquire and prepare the necessary test tools; ensure that the test team receives relevant training.

5. Perform testing

* Conduct appropriate quality reviews, such as code reviews and walk-throughs.

* Conduct testing, whether at the unit, integration, system, or use-acceptance level.

* Execute tests and collate test results; record results and incidents in the defect-tracking system.

* Analyze test results; monitor areas with a high incidence of defects and defect severity.

* Track and manage defect resolution.

* Where appropriate, repeat the test and reuse test materials to expedite retesting.

* Stage a pre-live simulation period; test the system under the expected business usage and observe results.

* Manage the decision to go live based on testing results; consider if the system has achieved a release standard.

6. Monitor the deployed system

* Monitor system performance using relevant monitoring tools; use performance reporting to guide performance-tuning activities.

* Capture any residual defects; assess their priority and, if serious, consider a rectification plan.

* Gather feedback and problem reports from users; analyze these to identify future improvements.

7. Manage successive releases

* Capture and prioritize new business requirements; assess their impact on the system before committing to them.

* Fix outstanding and residual defects.

* Plan the system’s next release; agree to a release plan with relevant stakeholders.

* Ensure a smooth transition from the current release to the new release.

6. Web Testing Key Challenges

Applications that are e-commerce based, or even if they are static websites, are different, and yet similar, to traditional software applications. There are different challenges, both technical and non-technical, that apply to testing a web application.

Testers are still learning how to best test e-commerce applications, as most of these applications are business critical, and it is still a massive and growing marketplace. Millions of dollars have been spent on websites, and the investors expect success. Unfortunately, e-commerce history is filled with expensive failures. Some of which could have been avoided by better testing before the site was opened to the general public (Samaroo, Allott & Hambling, 1999).

6. 1 Scalability and performance

Websites offer new challenges to developers, as well as testers. Scalability and performance are two areas that are significant in the e-commerce and web applications space. However, when new technology is used to make web applications perform well and scalable, new testing methodologies have to be created along with those technologies. Performance is critical, and based on a study from the Newport Group, more than half the recently deployed transaction based web applications did not meet expectations for how many simultaneous users their applications could handle (Shea, 2000).

The nature of websites as a whole offer challenges to the testers to effectively test the sites. They tend to be built in a requirements-free, rapid-development environment, all done with limited management support

(Glass, 2000). This combination of factors makes it very difficult for the testers to fully test the system to a high degree of quality.

6. 2 Time to market

Time to market is also a factor, as the concept of “ internet-time” which is generally considered to be about four times faster than normal time, has served to cut the test time for web applications, which affects quality and completeness of tests as well (Hayes, 1996).

6. 3 Usability Test

Most e-commerce applications have their interface running inside a web browser such as Netscape Navigator or Microsoft Internet Explorer. The browser-specific graphical user interfaces (GUIs) (e. g. hyperlinks, forms and back buttons) together with the add-on GUIs (e. g. multimedia plug-ins like Macromedia Flash) have different behavior than those on Windows-based client-server applications. Old testing tools may not be able to test the usability of the application in these new GUIs.

6. 4 High demand for regression testing

The continuously changing of deployment environment also presents a problem to test team. Before production, the application must be tested in all target deployment environments. With the continuously update of web browsers, run-time systems and operating systems, test team should assure the compatibility of the software with these updates. Different browser configurations must also be tested because they will affect the display and

behavior of the application screens. Many test cases need to be repeated many times.

6. 5 Other factors

Other factors that are very important regarding web applications are security, availability, reliability and recoverability. Uptime requirements for web applications are far more stringent than for off-the-shelf/shrink-wrap software. People expect that websites are secure, and are available twenty-four hours a day, seven days per week. When they are not, the business suffers (MacIntosh & Strigel, 2000).

A successful web application can be summarized to as being: usable, secure, scaleable, reliable, maintainable and highly available (Samaroo et al, 1999). If a site does not meet these criteria, then it may run into problems, so all of these factors need to be tested, and that is a complex process.

To summarize the challenges of web testing, it can be said that the normal testing challenges exist, such as requirements quality, etc. but the impact is much higher, due to the critical nature of websites, e-commerce websites in particular. Additionally, there are other non-traditional testing efforts that are required for websites, such as performance testing, scalability, and others which add further challenges to a website testing team.

7. Testing methods

The test types are structured into a series of stages that address the risks most efficiently. The test process framework presented in Table 7. 1 is intended to assist testers in constructing their own test process when they

know the test types that will be used. The framework will help to construct the detailed test stages from the test types.

7. 1 Seven Categories of Tests

To address the risks of the e-business project, there are 24 distinct test types, each of which addresses a different risk area. The test types are grouped into seven main categories:

* Static;

* Web page integration;

* Function;

* Service;

* Usability;

* Security;

* Large-scale integration.

Test Type

Test Content

Static

* Content checking

* HTML validation

* Browser syntax

* Compatibility checking

* Visual browser validation

Web Page integration

* Link Checking

* Object load and timing

* Transaction verification

Functional

* Browser page testing

* Server-based component

* Testing

* Transaction link testing

* Application system testing

* Context testing

* Localization

* Configuration testing

Service

* Performance and stress Testing

* Reliability/failover testing

* Service management testing

Usability

* Collaborative usability

* Inspection

* Usability testing

* Web accessibility checking

Security

* Security assessment

* Penetration testing

Large-scale integration

* Systems integration testing

* Business integration testing

Table 7. 1 Test Process Framework

7. 2 Test types for Static/Dynamic, Automatic/Manual

Test types can be static or dynamic and also can be automatic or manually. For example, static tests are those relating to inspection, review, or automated static analysis of development deliverables.

The test types do not always fit into the traditional unit, integration, system, and acceptance test stages. So, a structure that has test stages that categorize the test types in a technical way:

* Desktop development testing (broadly, of software that executes under the control of the browser);

* Infrastructure testing (of what runs on the servers);

* System testing (of the complete system in isolation);

* Large-scale integration (with other systems and business processed);

* Postdeployment monitoring (of live sites, retaining automated tests and using external services)

7. 3 Static Testing

Static tests are those that do not involve executing software under test by inputting data and commands and comparing the software behavior with expected results. Static tests do involve inspections and reviews of project documentation, specifications, or code. Static analysis of code using automated tools also counts as a static test and is an effective way of detecting not only faults, but also poor programming practices. Table 7. 2 lists the static testing addresses.

Test Objective

Technique

Ensure that al text in pages is spelled correctly

Content checking

Demonstrate that printed pages are complete, legible, and usable

Content checking

Demonstrate that Web pages are usable at varying window sizes, screen resolutions, and font sizes.

Content checking

Verify that al text, messages, help screens, and information provided and accurate, concise, helpful, and understandable

Usability assessment

Verify that HTML on Web pages complies with the HTML standard

HTML validation

Verify that CSS files comply with CSS standard

HTML validation

Demonstrate that web pages appear and behave consistently across the browsers in scope

Browser syntax compatibility checking

Verify by inspection that the appearance of web pages in consistent across browsers in scope

Visual browser validation

Table 7. 2 lists the static testing addresses

7. 4 Web Page Integration testing

The listed in Table 7. 3 all relate to failures to that would be noticeable as soon as a user first navigates to a page, clicks in a link on button to navigate to another page, or invokes a server-based components. These faults are grouped together because they relate to the integration of HTML pages to embedded objects, linked pages, or server-based functionality. Collecting these faults together means that the testing required to address them can be performed on a single web pages component when the linked objects are available. As each component is developed and combined with its linked components the links to other pages and objects can be checked within a development environment. Essentially, this is an integration process, and the tests that accompany it are integration tests. One of the biggest benefits of these tests is that they can largely be automated.

Test Objective

Technique

Demonstrate that all objects (frame page HTML, images, sounds, applets) referenced by the pages are available and can load.

Link checking

Verify that links to on-site objects load correctly when clicked.

Link checking

Verify that links to off-site objects load correctly when clicked.

Link checking

Verify that objects are small enough to download within an acceptable time.

Object load and timing

Verify the following:

. Correct components are referenced;

. Component exists and can be executed;

. Data passed to the component is transferred correctly;

Transaction verification

Table 7. 3 List of Web Page Integration Testing

7. 5 Functional Testing

Functional tests demonstrate that software meets its functional requirements, which describe what the system must do. Nonfunctional requirements, by and large, describe how the system must deliver its functionality. (Is it fast? Is it usable? Is it secure?) Performance, usability, and security testing are therefore nonfunctional test types. Configuration testing

could also be regarded as a nonfunctional test, but for most practical purposes, configuration testing is dominated by functional regression tests using a variety of platforms and configurations.

7. 5. 1 Architecture and Components

Before we look at the test types themselves in more detail, it is worth spending a little time looking at how Web applications are normally built from components. The problem with architectures and components is that there are an almost infinite number of combinations and permutations of these architectures, and the technical detail required to understand and test all such components would be overwhelming. At that moment, we’ll focus specially on the common browser-based functionality and the components residing on Web servers that are the next layer in these architectures.

7. 5. 2 Test types and the objects under test

Figure 7. 4 presents a schematic of a typical architecture for a Web system. As with many larger, more complex Web sites, its four-tiered architecture comprises the following:

* First tier: client browser;

* Second tier: Web server(s) accessed through the Internet;

* Third tier: application or object server(s);

* Fourth tier:

* Database server;

* Other legacy systems;

* External interfaces (e. g., banks).

Figure 7. 4 Schematic detailing scope of functional test types

The diagram sets out the boundaries for four test types, each of which indicates the system components that are within scope. In the real world, systems can be much larger and more complex, but they can also be smaller and simpler. Table 7. 5 presents somewhat idealized descriptions of the test types described here and their scopes in order to put them into context. The scope of the test stages and the test types they include much less ordered and clear-cut.

Test Type

Scope

Browser testing

Covers the component executed under the control of the browser

Server component testing

Covers the components residing on the server and invoked by the browser components

Application system test

Covers the technical components specific to the application under test and excludes internal client legacy or external systems

Large-scale integration

Covers the entire technical architecture for the system being implemented

Table 7. 5 Scope of the Functional Test Types

7. 5. 3 Automated Test Execution

The test types dealing with functionality can all benefit from the use of automated test execution tools. In the server-based component testing section, we suggest that custom-built dummy HTML forms or test drivers can support this test activity, and these are described briefly.

It remains a sad fact that most test-running tools end up on the shelf. The promise of test automation is very high, but the achievement of this promise can be quite a challenge. At this point, we will say that test automation should be considered as early as test planning is undertaken.

7. 5. 4 Browser Page Testing

Browser page tests cover the objects that execute within the browser, but do not exercise the server-based components. These components typically include the following:

* JavaScript (or VBScript) code embedded within an HTML page that implements special effects, or field validation;

* Java applets that implement screen functionality or graphical output;

* ActiveX components that execute within the context of a Web page;

* Plug-ins such as Quicktime or RealPlayer.

7. 5. 5 Server-Based Component Testing

Server-based component testing covers the objects that execute on the server, but are initiated by Java applets, ActiveX components, or HTML forms-based user interactions on the browser. These components typically perform standard processes, such as the following:

* Security checking and validation;

* Business logic to perform database enquiries or updates;

* Product catalog searches;

* Order processing;

* Credit checking and payment processing.

Usually, the server-based forms handlers are written at the same time as the forms that invoke them; however, we have found two problems with testing server-based code using the user interface:

* In some projects, the user interface may actually be the last thing written, so server-based components cannot be tested completely until very late in the project.

* It is common for server-based components to be programmed to deal with a very broad range of input parameters, invoking complex transactions that interface with legacy systems. Large numbers of tests may be difficult to

implement manually, so an automated alternative can be more effective and economic.

7. 5. 6 Transaction Link Testing

Transaction link testing (link testing) aims to address the problem of integrating the complete end-to-end functionality to ensure that the entire transaction is processed correctly from the user action on the browser interface to the back-end systems. Link testing focuses very much on selected test cases that exercise particular interfaces between components in the technical architecture to ensure that mismatches between components are minimized.

7. 5. 7 Application System Testing

Application system testing aims to cover functional testing of the application as a whole. Where a supplier providing the system is unable to build a complete test environment that includes, for example, the client’s legacy systems and interfaces to banks or other partners, system testing would take place in an environment with client-supplied interfaces stubbed out. Testing of the complete technical architecture is covered later in large-scale integration testing and is likely to be a key part of acceptance testing. Application system testing in the form described here is most likely to be performed by the suppliers of the system itself in an environment that allows the system to be tested in isolation from its external interfaces.

7. 5. 8 Context Testing

There are several risks relating to the context (or, perhaps more correctly the loss of context) of Web transactions. One of the differences between client/ server and Web applications is that for home-based users, the network connection is under the user’s control. Dial-up connections are usually temporary and are generally less reliable than the permanent high-speed LAN connections most office-based workers use. Connection, disconnection, reconnection anomalies cause serious disruption to most users when they access the Web. The behavior of Web sites under these scenarios varies widely across Web sites, and in many cases, poor design causes Web site owners and their users dismay.

.

Unlike with client/server applications, the user accesses a Web application through a browser. The browser itself has buttons and controls that allow the user to interfere with the normal operation of the Web site in the following ways:

* The back button allows the user to go back to a previously displayed Web page cached in the browser’s memory. If that page is used to execute a purchase with a one-time discount, the transaction could be re-executed many times with the discount applied. To avoid this, the developer can make these pages expire immediately so that the browser cannot just restore them from its cache.

* Refresh (IE) and reload (Netscape) buttons allow users to re-execute transactions that produce the current Web page. Again, this could allow

users to repeat a system transaction without having to repeat previous pages.

* The history facilities in browsers bookmark previously visited pages and navigate directly back to them, which re-executes the transaction that loaded the page. The application Web pages could therefore, in principle, be re-executed in any order the user wished, which could wreak havoc with the application.

7. 5. 9 Localization Testing

Multilingual sites are often built with all language-specific material codified into a database. In this case, localization testing is all about verifying that all user messages, prompts, and output are translated correctly and that the functionality delivered to the end user is identical. Checking translations isn’t quite as simple as comparing text translation with a foreign dictionary. Have you ever seen the names of dishes on a foreign restaurant’s menu translated literally? The people checking translations should speak the target language as their mother tongue so you can be sure that the translations are accurate, not just literal.

7. 5. 10 Configuration Testing

Configuration testing aims to demonstrate that your Web application will operate correctly on your nominated range of client hardware, OS, and browser combinations. On the Internet, you have no control over the end users’ platform, so to avoid problems later, it is best to test a range of configurations to ensure that you can support at least the most common

combination. The problem, of course, is that there is a virtually unlimited number of possible combination, Table 7. 6 provide, some indication of the possibilities that face the tester.

Most of the component types listed in Table 7. 6 have multiple versions. The number of combinations and permutations are in the hundreds, or even thousands. The essential difficulty of configuration testing is therefore deciding on the scope of the testing. The big questions include the following:

* What do your users actually use?

* Which configurations should you support? (Not all will be worth considering)

* Which can you test economically?

* Which can you test automatically?

OS platforms

DOS, Windows 3. 1, 95, 98, NT 3. 51, NT 4, 2000, Xp, Macintosh, Linux, Unix

Network connections

Dial-up modems, direct Internet lines, cable modems, ADSL. wireless

Commercial services

MSN, AOL. and others

Browsers

IE, Netscape, Opera, and many others-too many to mention.

Browser versions

All browsers have several versions perhaps with country-specific

Table 7. 6 Common Configuration Options

7. 6 Service Testing

Companies build e-business Web sites to provide a service to customers. In many cases, these sites also have to make money. If a Web site provides poor service, customers will stop using it and find an alternative. Quality of service as provided by a Web site could be defined to include such attributes as functionality, performance, reliability, usability, security, and so on. For our purposes, however, we are separating out three particular Web service objectives that come under the scrutiny of what we will call “ Service Testing”:

* Performance: The Web site must be responsive to users while supporting the loads imposed upon it.

* Reliability: The Web site must be reliable or continue to provide a service even when a failure occurs if it is designed to be resilient to failure.

* Manageability: The Web site must be capable of being managed, configured, and changed without a degradation of service notice able to end users.

7. 6. 1 What Is Performance Testing?

Performance testing normally requires a team of people to help the testers. These are the technical architects, server administrators, network administrators, developers, and database designers and administrators. These technical experts are qualified to analyze the statistics generated by the resource monitoring tools and judge how best to adjust the application or to tune or upgrade the system. If you are the tester, unless you are a particular expert in these fields yourself, don’t be tempted to pretend that you can interpret these statistics and make tuning and optimization decisions. It is essential to involve these experts early in the project to get their advice and commitment and, later, during testing, to ensure that bottlenecks are identified and resolved.

7. 6. 2 Reliability/Failover Testing

Assuring the continuous availability of a Web service may be a key objective of your project. Reliability testing helps to flush out obscure faults that cause unexpected failures so they can be fixed. Failover testing helps to ensure that the measures designed for anticipated failures actually work.

7. 6. 3 Service Management Testing

When the Web site is deployed in production, it has to be managed. Keeping a site up and running requires that it be monitored, upgraded, backed up, and fixed quickly when things go wrong. The procedures that Web site managers use to perform upgrades, backups, releases, and restorations from failures are critical to providing a reliable service, so they need testing, particularly if the site will undergo rapid change after deployment.

The management procedures fall into five broad categories:

* System shutdown and start-up procedures;

* Server, network, and software infrastructure and application code installation and upgrades;

* Server, network, and software infrastructure configuration and security changes;

* Normal system backups;

* System restoration (from various modes of failure) procedures;

The first four procedures are the routine, day-to-day procedures. The last one, system restoration, is less common and is very much the exception, as they deal with failures in the technical environment where more or less infrastructure may be out of service.

7. 7 Usability Assessment

Put an unusable site onto the Web, and your users will not stick around very long. Your users have the freedom to abandon your site and visit your competitors’. It may have been possible to implement difficult-to-use systems in the past, but it is no longer. One of the key selling points for Web sites is their ease of use. A usability fault is a potential problem in the appearance or organization of a system that makes it less easy for users to use. Operationally, a usability fault is any clear or evident violation of an established usability principle or guideline; or it is any aspect of a system

that is likely to lead to confusion, error, delay, or failure to complete some task on the part of the user.

Usability faults have two dimensions:

* The location and identity of a fault;

* The problem and rationale for identifying it.

Usability and usability assessment have never been as important as they are now. Typical Web site users have low boredom, frustration, inconvenience, and insecurity thresholds. Home-based users may never have been trained in the use of a computer, let alone browsers and your Web applications. Regardless of the user’s level of sophistication, if your Web application doesn’t allow enquiries to be made and orders to be placed easily, quickly, and reliably, the site will fail. If the user cannot easily understand from your Web site how to proceed, the user will leave your site and go elsewhere.

7. 7. 1 Collaborative Usability Inspection

Collaborative usability inspection is a systematic examination of a finished product, design, or prototype from the point of view of its usability by intended end users. The process is a team effort that includes developers, end users, application or domain experts, and usability specialists working in collaboration. A major benefit of the method is its own usability: It is easy to learn. Experienced inspection teams have been known to detect 100 usability faults per hour. The collaborative nature of these inspections means that users and developers understand the relationship between user interactions and design constraints and decisions. These inspections can be

performed at any stage of development from the assessment of prototypes to the finished Web site, but, of course, the cost of fault correction increases the later the inspections take place.

7. 7. 2 Heuristics on Which to Base Inspections

Inspections are guided by a set of rules or heuristics for good user-interface design. These are used as a framework for identifying and categorizing usability faults.

Nielsen promotes a (popular) set of 10 Web design heuristics summarized in Table 7. 7. His book, Designing Web Usability, is packed with usability design guidelines from which you can extract your own preferred rules if you care to.

7. 7. 3 Usability Testing

Usability testing aims to detect usability faults in a system by using the system to perform selected tasks. There are two broad ways of performing usability testing: in a usability laboratory or in the field. We will only consider field usability testing here.

Usability testing of a finished system is effective for finding certain faults, but is no substitute for good usability design. Experience shows that usability tests uncover fewer faults than inspections, but the faults that are found are often more subtle or unusual in nature, and they can be more serious. Usability testing requires that a working system be available, so these tests tend to be possible only late in the project. If there isn’t time to fix these faults, they may be left in the released product. Overall, usability tests are

most effective when used to investigate specific areas of an application or to explore particular issues.

7. 7. 4 Web Accessibility Checking

It is possible to conduct manual inspections of a system to detect departures from user-interface standards or accessibility guidelines. In the context of the Web, accessibility refers to the ease with which anyone can make use of the Web, regardless of his or her technology, location, or disability. The most important accessibility guidelines for Web-based systems are defined as part of the Web Accessibility Initiative (WAI) – the “ Web Content Accessibility Guidelines”. This document defines a comprehensive collection of guidelines to make your Web pages more accessible to users with disabilities, although the vast majority of guidelines would make a site more accessible (or usable) to all users.

There are several assistive technologies now available to help users with disabilities use the Web. These include the following:

* Text-to-speech (TTS) browsers that use synthesized speech to read text on Web sites to a user;

* Text-only browsers that render Web sites in a text-only format;

* Voice-enabled browsers that navigate Web sites using speech commands;

* Specialized keyboards and mice;

* Voice-recognition software;

* Refreshable braille devices that transfer text onto a special device;

* Screen magnification or enhancement software.

7. 8 Security Testing

When most people think of security (and the challenge that hackers present to it), they have a vision of longhaired college dropouts working into the wee hours at a furious pace, trying to crack a remote system. By guessing passwords and through perseverance, luck, and ingenuity, they break into the bank, government department, or evil mastermind’s mainframe system. Although this image works well for the movies, it isn’t helpful to our understanding of the security threats that face e-business systems. Determined attackers often work in teams, they work almost entirely with automated tools, and they may wait patiently for months before attacking perhaps hundreds of sites in a single day. Many of them adopt a scattergun approach to acquiring targets, but they really are more sophisticated than the movies usually represent them.

Although outsiders are a serious threat, it should be borne in mind that insiders perform many break-ins. Whatever their motives (frustration, anger, or a grudge against an employer), insiders are particularly dangerous. If they are still employed, not only do they have access to your systems, they probably also know your network topology and the servers that host critical services and are trusted within your organization – what they don’t know, they can find our easily. Even if they have left your company, their knowledge gives them a head start in their intrusion efforts. Either way, your

internal security policies should be regularly reviewed and thoroughly implemented.

7. 8. 1 Security Assessment

A security assessment is a review of a site’s hardware and software configuration to identify security vulnerabilities. The assessment team works with the full cooperation of the technical staff that designed, implemented, and supported the site to be assessed. Security assessments, being systematic attempts to identify all security weak points, are the most cost-effective way to address security concerns. The structure and definition of an assessment varies with the scale of the project and objectives of the organization that commission it. Assessments can be conducted from an internal or external point of view.

Assessments can focus on servers, networks, or firewalls and can be technically or non technically oriented.

7. 8. 2 Penetration Testing

Penetration tests aim to demonstrate that within a short period an intrusion can be achieved and that a system is vulnerable to attack. Most attempted Internet attacks are performed by uninventive script kiddies. There are a wide range of capabilities between script kiddies (who know little) and determined attackers who have in-depth networking experience and programming knowledge. Somewhere in between, there are the casual hackers. These are probably the main threat to Web sites. Typically, a determined attacker will spend up to 3 days trying to crack a site, so

penetration testers are usually given approximately 3 days to achieve an exploit. If the test does not expose a vulnerability, one can reasonably assume the site is secure from this kind of attacker (assuming your testers are as competent as the attacker, of course).

7. 9 Large-Scale Integration Testing

The risks associated with Large-Scale Integration are the same as those we tackled using transaction link testing, but the interfaces we are concerned with here exist at a higher level between multiple systems and between systems and the business process. Integration is an often misunderstood concept because the integration process starts almost as soon as coding begins.

In an e-commerce application the scope of LSI testing might cover integration with external banks or credit-card processing systems, product wholesalers or distributors, as well as internal legacy systems.

The notion of fit is appropriate for component-to-component integration, system-to-system integration and system(s)-to-business-process integration. We have conducted business integration testing (BIT) on many client projects where user acceptance was (at least partially) based on the results of these tests. Using the same integration framework for user acceptance makes test planning easier, and business management will support this activity because they can understand how it will give them confidence that the delivered service will work.

We are not suggesting that BIT is the same as user acceptance testing. Rather, we expect that in many organizations, the final stages of LSI testing provide some confidence to users that the system will work as they require. If you are asked to plan an acceptance test for your users, we expect that the techniques we describe in BIT in particular will play some part in your planning.

We separate LSI testing into two stages because the risks and test objectives differ. Systems integration testing (SIT) is more technically oriented because it is at this point that the physical connections between systems are established and used for the first time. The tests are somewhat more white-box-oriented in that the physical interfaces between systems must be understood enough to allow test designers to prepare tests that cover them. BIT is more focused on the paths through business processes to ensure that the integrated systems provide seamless support to the user activity throughout.

7. 9. 1 Integration Analysis

In complex environments where a large number of interfaces are to be tested, a systematic approach to the identification of interfaces and the transactions that exercise them was essential. Because the integration knowledge is extracted from many sources, it is a good idea to get your integration inventories reviewed by the development project teams. There are three techniques for documenting integration knowledge:

* Identification of systems and system-to-system interfaces and the compilation of integration inventories;

* Identification and documentation of high-level business transactions that are dependent on these interfaces or trigger their execution;

* Analysis of the use of data across multiple systems for the purpose of tracking data flows and reconciliation.

Planning and preparing an LSI test that addresses the risks of concern normally require information from all three dimensions: interfaces, transactions, and reconciliations. The integration analysis takes two views of systems and interfaces:

* A technical or system view (to prepare for SIT).

* A business view (to prepare for BIT).

The system view and the business view are similar in approach: The system view captures interfaces and transactions in terms of physical connections (files or messages) and system transactions; the business view captures interfaces in terms of business entities being transferred between applications via business transactions.

7. 9. 2 SIT

To plan SIT, the tester needs to know quite a lot about the physical interfaces between systems. Only by knowing the following about the internals can the tester design tests that will exercise these interfaces adequately:

* The details of the internals of the interface;

* The nature of the system-to-system dialogs;

* How to exercise the interface from the application user interface or batch processes;

* How to create test data to exercise the interface;

* How to find evidence that the interface works.

We said earlier that one of the problems of LSI testing is that it can be difficult to find details of interfaces. It’s not just the interface details that cause problems. It may be that there is no documentation available at all for the legacy systems. Once the integration inventories are prepared, the tester follows a simple process to define the integration test:

* For each interface, identify the dialogs between systems and which business or system events trigger them to work.

* Derive test cases for success and failure to negotiate each step in the dialog.

* Derive test cases from the interface data validation and use descriptions to ensure that valid data is transmitted, invalid data is rejected, and the storage and use of data in each interfacing system reconcile.

* Define your test environment and infrastructure needs early, so they are met in good time.

Integration tests tend to fall into one of two types: They are either very simple (and easily automated) or they are very complicated and have to be

executed manually. When the tests are executed, early tests focus on the correctness of the interface calls. Later tests (usually automated) focus on memory leaks, loss of synchronization between systems, and failure and recovery of clients, servers, or the network.

7. 9. 3 BIT

The primary aim of BIT is to provide final confirmation that systems, processes, and people work as an integrated whole to meet an organization’s objectives and provide a sophisticated, efficient service to its customers. BIT takes a process- and people-oriented view of the entire system.

BIT objectives can span a large range of issues. Perhaps a new process needs to have rough edges removed; perhaps an existing process needs changing to reflect a new way of doing business. Alternatively, the training provided to end users might be the problem: perhaps users need more detailed instruction in how to use particular aspects of the system. BIT is a more rounded approach to finding faults in system implementations, not just the software.

BIT differs from SIT in that it is more likely to be associated with user acceptance. Assuming that the technical testers have demonstrated that the interfaces between the new system and other systems work in SIT, the imperative for a business wishing to deploy the new system is to ensure the system supports the intended business activity. For example, if a system supports the on-line purchase and delivery of books, the following questions must be addressed:

* Can a customer search for a book, add it to a shopping basket, and place an order?

* Can a customer’s credit card be validated and can payment be authorized and processed successfully?

* Does the legacy order processing system receive the on-line order accurately?

* Is the book in stock, located in the warehouse, packed, labeled, and dispatched correctly?

* Are order confirmation, progress notification, and thank-you e-mail messages sent at the right time? Do they reliably reach the Customer?

* Are payments processed promptly, accurately, and reliably?

Ultimately, the sponsors of the system want to know whether the new system meets the cardinal business objectives of the project. Testers must develop and execute selected business scenarios that will exercise the integrated systems to provide evidence that they support the business process in its entirety.

Compared with SIT testing, BIT may take a smaller number of test cases to give confidence that the system works correctly because the number of tests should be limited to a finite number of business scenarios. However, BIT alone may not provide enough information on which to base acceptance of a system. Users will certainly want to see many more system transactions executed to ensure that the core functionality of the new Web system works.

8. Other testing considerations

8. 1 Content analysis

The actual content provided by a site also needs to be “ tested.”

Offensive, misleading, and litigious content. An e-commerce site’s contents must be sound. In the UK, legislation such as the Trade Marks Action (1994), Control of Misleading Advertisements Regulations (1988), and Obscene Publications Act (1976) applies to all published material, including Web content. Products or services that are incorrectly described or misleading (check those product descriptions!) may violate the Trade Description Act.

Infringements. Test your Web site content for infringements. Graphics and images used in the design should be royalty free (if not wholly owned); also check for copyright infringement. In some circumstances, the courts will consider use of trademarked material without the owner’s written consent as infringement. Courts would almost certainly view discrediting or denigrating a competitor’s brand as an infringement.

Personalization. Increasingly, e-commerce systems are incorporating customer relationship management functionality that provides personalization for individual customers. In many cases, a personalization profile determines what content the site offers to these customers. However, situations can occur where the site offers incorrect or inappropriate content to customers because of errors in the personalization functionality.

8. 2 Availability

Unavailability equals lost revenue; it also harms a business’s reputation and can encourage customers to take their business to competitors. Businesses need to offer 24/7 availability to their customers, with availability levels of 99 percent or higher being the desired norm.

Unacceptable levels of unavailability. Several factors can influence availability, such as hardware reliability, software reliability, the effectiveness of load balancing, and the database’s ability to handle concurrent users. Before going live, predicted business usage patterns should indicate maximum stress levels. You should test system availability against the maximum stress levels plus a safety margin for a defined period of time.

Denial of service. Yahoo, Amazon. com, and Buy. com have all recently suffered denial-of-service attacks. Such attacks (also known as saturation attacks) involve bombarding the Web server with bogus requests, making it inaccessible to genuine users. Organizations are not defenseless against such attacks; Amazon. com for example, uses security software to filter out bogus requests. Test your e-commerce systems for vulnerability to denial-of-service attacks.

8. 3 Backup and recovery

You can’t guarantee that any component of your e-commerce system won’t fail, whether hardware or software, so it’s sensible to test that you can quickly recover from a failure when it does happen.

Failure/fall-over recovery. Design systems so that one component’s failure doesn’t necessarily bring down other components. In MS Internet Information Server (version 4. 0), for example, Active Server Page scripts can run from the Web server as separate processes.

Test both the speed and ease at which systems can recover from component failure; for example, how do you