Design for Testability of Web Applications – Manager's Perspective

Raghav S. Nandyal
Chief Executive Officer

SITARA Technologies Pvt. Ltd.
3-6-460 'Gokul Kunj', #304 Street No. 5 Himayatnagar Hyderabad AP - 500 029
Email: raghav_nandyal@sitaratech.com
URL: http://www.sitaratech.com

Abstract:

When the web was conceived, it was primarily designed with capability to display hypertext with inline images with the capability of linking pages together. Today, web applications are designed to serve up dynamic content, perform validation of input data in forms and process this information using databases external to the web server. And, form processing which uses CGI programming makes use of both compiled and interpreted languages that come in multiple varieties. Testing of web applications is an entirely different ball game altogether requiring a very learned team that is capable of debugging both complied languages such as C, C++, Perl, Java, and Visual Basic and interpreted languages such as TCL, shell scripts and Javascript. A manager's perspective in determining and selecting from the plethora of choices plays a very crucial role. The paper provides an understanding of how to manage and handle testing of web applications using good design principles.

1.0 INTRODUCTION

The content that is served up by web applications is truly multimedia with capacities extending to collecting data, reading and writing information using other applications. The outcome of this capacity is that information is literally computed real-time using multiple choices of data that information can take. Now, here is where the traditional concepts of establishing a lower bound and an upper bound for data for boundary tests are simply not feasible. Testing of interactive, dynamic data-driven and information-rich web applications is not a mere extension of traditional testing paradigms.

The primary difference between a traditional GUI based application and one that is browser-based is that features have the potential to be exercised in multiple sessions. This is a useful point to note while ensuring "**isolation of features**".

From an end-user perspective web applications tend to develop a sequential pattern of execution. In order to enhance the web experience, it is prudent to make sure that the number of sequential screen transitions achieving an objective from a feature execution be at a maximum of "**3 navigation units**". It is useful to point out at the outset that the 3 navigation units rule lends well to a "star-topology site design" as found in [1]. This point will be elaborated further in the paper under section 3.0.

Form data elements are primarily of two types. One type of data is what is entered physically, and the other type of data is the dynamically populated form elements computed based on choices made on other data elements of the form. There is also a big blessing in separating the production or staging environment (web-server and the database) and the application environment. So, confidence testing requires "**separation of data types**" to confirm overall results from form processing.

The above 3 concepts together with other ideas will be described in the paper after well tried-out application on numerous testing exercises.

2.0 ISOLATION OF FEATURES

The significant departure from traditional testing techniques and testing web applications come from the mere explosion of domain knowledge that becomes necessary for a test engineer. An in depth understanding of client-server architecture, network security and restrictions, behavior of application components that might be used such as ActiveX or Java Beans, peculiarities of scripting methods such as HTML, Dynamic HTML, XML and client side versus server side processing (ASP, JSP) of the HTML page itself make the role of planning the testing process a very delicate job. This domain knowledge is essential for designing applications while keeping isolation of features in mind.

Speaking at a very broad level, the test plan must consider effective separation of the following:

1. The User Interface testing
2. Database testing and database population methods
3. Network security testing

Much of web application testing is still a manual process. Web rendering of HTML code is a matter of checking to see how each page (static and dynamic) gets displayed on the most popular clients (Internet Explorer or Netscape Navigator). Applications that leverage these thin-clients to serve, as the front-end GUI, must be tested for rendering on both these clients. A useful design decision to make is to go for a Star topology of the site and to restrict the number of navigation units to at most 3 units from the hub being the index page or the homepage.

3.0 3 NAVIGATION UNITS

A good rule of thumb to use while designing web applications is to minimize the depth of navigation to a maximum of 3 units. The primary reason for imposing this constraint is that web applications containing either static or dynamic pages are intended for maximum usage – in other words better user-experience. It would almost become a nightmare to navigate, test and validate an ill-conceived site with numerous pages and navigation links. One of the best strategies is to use a "Star" navigation style [1]. The Star navigation has a central hub that is the index or the default page that gets loaded upon a URL (Uniform Resource Locater) or URI (Uniform Resource Identifier) invocation. Radiating from the central hub are the different features of the application or the features of the site. It is useful to note here that in this style, the number of radiating features or functions can be unlimited and is an extensible design concept. The binding of the 3 navigation units is on each one of these radiating spokes. The hub or the index page is only 3 clicks away – always! Designing applications and sites in the "Star-Site Topology" has been found to contribute tremendously to a reduction in time spent on both system and integration testing effort. Using forms to capture information and rendering the validation along with the form data or information processing is ideally a 3-step process again. First with the user providing the necessary data or information on the form. Second, script validation of field data for integrity. And finally execute the form information on a web-server and, return of appropriately processed information to the user. So, making a design decision to restrict the navigation units to 3 units is not going to hinder form processing in any way. Star-site topology adds to the benefits of extendibility and scalability of the site architecture. The emphasis in this point is to ensure modularity of functions placed on the radiating hubs representing the features of functions of the application or the site. When applications such as financial processing systems [2] need to be developed, it is useful to separate the form entry, processing and subsequent transactions into two separate partitions or identities. The 3-navigation units rule can now be applied to each one of these two separate partitions or identities. There are many advantages with this partitioning. Should you want to either expire a page (in secure http transactions using https protocol) or if you want to display forms with related user inputs, they can easily be isolated into two separate functions one set of transactions based on the results from the previous set of inputs from the main hub. The benefit with this design is that a sophisticated (secure) system design can be limited to just two browser windows. Most designs that have been ill conceived and have the further disadvantage of poorly lending themselves to testability are seen to spawn browser sessions in gay abandon. It is both frustrating to test and also a frustrating user experience! This partitioning that is suggested of features into 2 entities at the most leads into the next topic, which is "separation of data types". Many secure applications have to be conceived with built-in timeouts for non-usage in a session.

It is easy to ensure end-user security and protection by closing a connection to a secure system from a partition that deals with features having a need for secure transactions. Most times, the partition that handles the not-so-secure information processing can stay alive without the user having to go through with entering all of the information again in a time-out based application. This is a useful feature to note for providing a better user-experience on a website.

4.0 SEPARATION OF DATA TYPES

The primary advantage of separation of data types into a 2-partition application design with a 3-navigation units rule on each one of these partitions as in [2] is to render isolation of test data attributes. This is a useful feature while building secure applications and transactions using the https protocol. Design of the form itself must ensure that the different data types such as combo boxes, radio buttons, check boxes, text areas and text entries are properly conceived with adequate properties to withstand end-user usage. For example, text entry is normally used to provide short input strings and is restricted by the MAXLENGTH tag in the definition. However, it is more than likely that end-users enter large buffers full of data into a text entry. The web-application must have the capability to handle such extremes without crashing. For example: when you expect the user to type a 6 digit code, but he enters a long string with more than a few 100 characters instead, how should the application handle this without crashing the web server? It is always useful to test using extreme testing strategies while an application is still in production keeping ignorant or even malicious user perspectives.

Testing web-applications is not restricted to only testing the GUI from the browser. One has to write queries and tweak the database itself to figure out the outcome of data processing. The role of the test team has more to do with deciding a testable database design using script based verification and validation procedures. A normal tendency in web-application development is to have the commissioning of web-applications separated into 2 stages. CGI scripts are normally used in developing applications. Development servers are used to develop and test applications before production runs of the application are ready to be deployed on a production server. CGI scripts are simple programs that have to be tested in a safe environment. It is important to shield scripts from the rest of the web content. Data separation also entails separating the test data from the data that will be used in a real-life scenario that is more idealistic. Separation of the test data is also useful to prevent loss of information while testing scripts that are going to permanently replace data on the database system. Duplicating and testing the scripts on copies of real-life data is a better way to manage testing of CGI scripts than risking loss of data permanently! Development servers are not part of the Internet and are also called the staging servers.

5.0 MISCELLENEOUS POINTS

Command line debugging of scripts is a recommended way to debug instead of accessing them from a web server. Those of us who are schooled with Unix, as background will immediately acknowledge the importance and challenges in debugging from command lines! It is much faster and easier to catch syntax errors in scripts when they are executed from command lines. Scripts written using Perl have flags (-w) to warn about code that could be potentially dangerous. The w flag is saving grace. Testing scripts from command line have the benefit of displaying the output on the console rather than directly sending it to the Web server. HTML that is being generated by the script can be validated for correctness. After the scripts are tested from the command line and are ensured to provide the intended features, testing using the web server is advised. These scripts are now "trusted scripts". One other way to test scripts is to use copious quantities of print statements to throw actual logs of execution details.

Irregular execution of trusted scripts through a web server is normally due to either inadequate permissions when a 403 Forbidden message is indicated or a 500 Server Error. A 500 Server Error is encountered when the server expected to receive something from a script and it did not get it. Normally, the server error log will contain the actual error recording. This log is normally under the home directory of the server and can be examined as a text file. Server Misconfiguration errors are due to a failure of the script to include a blank line between the end of the header and the beginning of the actual content. HTTP requires a blank line be included between the header and the content to indicate where the header ends and where the content begins.

When script errors occur, the server normally provides feedback instantaneously. Logic errors are harder to track down and fix. It has been useful to use the 3-navigations Star topology rule with a 2-partition engagement for application design. Java server pages behave in a peculiar manner when you interrupt their full execution by trying to execute options that are displayed partially as the page is getting loaded. Run time errors occur which are otherwise not harmful. The solution for run time errors is to let the ".jsp" file execute completely. And last but not the least, when you make changes to an html or an ASP file that generates html and put it on the staging server, don't forget to refresh or re-load the page! You might have made the changes to the html, but in order for it to render on the browser a refresh is a must. When dealing with numerous changes while testing, it is not uncommon to forget to do this and end up spending time looking and re-looking the html endlessly.

REFERENCES:

[1] Study of site design of www.sitaratech.com
[2] Study of site design of www.citibank.com
[3] Software Reliability Engineering, IEEE Computer Society Press. McGraw-Hill. 1996