Chapter XII
Testing for Web Applications

David L. Mills
University of Memphis, USA

ABSTRACT

This chapter explores the concepts and challenges behind testing Web applications, and explores the latest testing techniques and best practices. As our reliance on the Internet grows, the quality and reliability of online resources become critical. Unfortunately, significant research shows that the current approaches to modern Web development are woefully inadequate. It is important that there are processes in place and best practices established to ensure that the development of Web applications can take place with an assurance of quality. In addition to offering an initiation to some of the modern testing methods and tools, the authors hope to motivate readers to consider testing as a multi-purpose tool to be used throughout all stages of development.

INTRODUCTION

Since its inception the Internet has been expanding in size, pervasiveness, and functionality with astonishing speed. It has become a fundamental tool in business, government, and education, not to mention our personal lives. In order to support growing demands, the Web has evolved from static pages of information to complex applications with functionality equivalent to that of modern software. As we grow more dependent on online resources, their quality and reliability become critically important (Hendrickson, 2002). Unfortunately, research suggests that practices to ensure these qualities are generally weak or missing from Web development projects.

This chapter aims to introduce the fundamental concepts of testing Web applications, and to provide insight about current best practices and resources. Not only will this chapter inform readers about the challenges of Web testing and the tools/techniques available, but it should also
provide motivation for software developers to integrate a sound testing solution into their software development life cycle model(s).

**Concepts in Conventional Testing**

Software testing is a method of quality assurance involving the analysis of software to evaluate features and ensure that the requirements are met. The tasks involved in testing are aimed at verification and validation (IEEE, 1990). The goal of verification is to make sure the product conforms to specifications. It is the process behind the question “are we building the product right?” Some sample verification tasks include inspections, reviews, and testing. Alternatively, validation is aimed at making sure that product build meets the needs of the user. It is the process behind the question “are we building the right product?” Evaluations and customer testing are typical validation tasks.

Testing techniques can be categorized as either black-box or white-box. Black-box testing takes place from an external view of the system, meaning that it is carried out without access to the code or knowledge about the internals of the program. The focus of such testing is on the functionality of the program. White-box testing, which is also called structural or glass-box testing, is performed with an internal view of the system. Tests are generated using the code as a guide (Myers, 2004).

There are various types of tests, each having a different focus and taking place at a different level. Unit testing targets individual components of a program. This usually refers to individual methods and/or classes (IEEE, 1990; Myers, 2004). Unit tests evaluate modules independently of one another. Integration testing is performed on combined components of the system, and focuses on their interaction. System testing is a form of black-box testing in which the system as a whole is tested. These tests can target functional or non-functional specifications.

Functional testing involves test cases that were created based on functional specifications, as opposed to non-functional testing, which is aimed at features such as speed and ease of use. Some common non-functional tests include performance testing, stress testing, and usability testing. Performance testing is evaluating the system compliance with performance specifications. Stress testing evaluates the behavior system when it is pushed with a heavy load to the limits of performance. Usability testing assesses the ability of users to learn to operate a system. This type of testing is often performed by specialists that observe humans interacting with the system.

Acceptance (aka Customer) testing is a form of black-box testing that defines the criteria that a system must meet to be acceptable for delivery, that is, a way of validating the requirements. This is a powerful and beneficial tool. It enables both the customer and developer to ascertain what work is complete and what work remains at any point, and it is way for the developer to validate that the customer is satisfied (Jeffries, et. al., 2001). Since the acceptance tests serve as a testable version of the critical requirements, when all the acceptance tests for a system pass, the system can be considered complete. Acceptance tests rarely involve code at the specification level, but they should be clear and understandable to both the customer and the developer. Collaboration is acceptable, even encouraged, but it is critical that the customer is always directly involved in writing the test cases. To facilitate their communication, acceptance tests are usually provided in the form of spreadsheets, tables or scripts that can be executed directly or as automated tests.

Regression testing is the re-execution of previous tests created for a system or component. They are typically run after significant changes are made, in order to ensure that intended functionality was not lost. Many types of tests can be included in the regression test suite, including unit, integration, system, and acceptance tests; though only a select set of tests is usually included as it