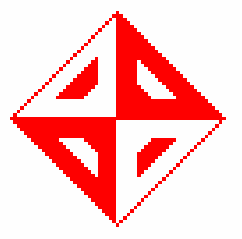
Software Test Specification

April 23rd, 2006

SmartGuardian

SmartGuardian™ Software



Computer Engineering Department
Middle East Technical University

by

SmartSoft Network Solutions, Inc.

Tayfun Şen - Özden Meren - Sedat Zelyüt - Hakan Özadam - Akın Öztürk

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose of the Test Specification Document

The purpose of the Test Specification Report is to formally describe the testing process that plays an important role in software development. This document will be used as a guide in testing the software product in both performance and feature aspects. The test specification will be used as a guide in determining the performance of the software and comparing the results with the formal expectations and requirements as has been specified previously.

1.2 Goals and Objectives

The ultimate goal of the test specification is to ensure that the software product is as reliable as it could be and to make sure that as much requirements as possible are met. The end-product is expected to be free of trivial bugs and it is expected to work as described in the user guide to be written. Of course, all software has bugs, it is a utopia to have bug-free software, but the objective of this document is to help developers fix most of the bugs using systematic and formal methods as described in this document.

1.3 Scope of the Document

The scope of the document covers the testing procedures to be used and the testing methodology is covered for each module. Software testing described in this document is twofold: black-box testing is used when testing the system as a whole without the internal structure of the system in consideration and white-box testing when the internal logic of the software, ie. in procedure atomicity, is used in testing.

The test specification document is intended to be a living document. The results of the tests conducted according to this document will be kept logged and reserved for

reference. These results will also be used in the software feature documents that will be used in advertising the final product. The test results will be used when evaluating the "success" of the project and the software as a whole.

1.4 Testing Constraints

There are many constraints related with the testing of the software product. The major constraints are the time constraints, personnel, performance and platform constraints.

1.4.1 Time Constraints

The testing is to start after the first release and go until the final release. Also, software maintenance term will contain testing using the feedback from the users of the software. The total time for the formal testing before the final release is about 1 month. This time has to be used efficiently to make most use of the testing specifications.

1.4.2 Personnel Constraints

The SmartGuardian Network Solutions is a small company with a staff of 5 persons. The small number of persons both presents an advantage and a disadvantage. The advantage of being a small company is that the decisions are taken quickly, the company becomes agile and development related tasks are determined in a quick way. The disadvantage is obviously less programming power to manage the project. The limited number of man-months resulting from small number of staff has to be managed efficiently. As SmartSoft, every member is responsible for the testing of the modules he has contributed to and also other modules that are in the system.

1.4.3 Platform Constraints

The SmartGuardian software is primarily developed to work in the Linux systems. However, many flavors of Linux OS exist in the market with many differences. The

idea of the platform interoperability is to have SmartGuardian work with different operating systems. We have used Gnome tools to achieve this platform independence. While installing our software, checks are made to detect environment variables and compiler features. This will ensure that SmartGuardian compiles works correctly with operating systems other than Fedora Core v4 which is the development platform.

Other side of the interoperability is the support for different browsers. Our system is expected to work with wide range of browsers and web servers. The system will be tested with different web browsers such as Firefox, Internet Explorer and different web servers such as Apache, MS IIS etc.

1.4.4 Performance Constraints

According to the literature survey we have conducted at the beginning of the previous term, most of the companies in the technopark area have 30 or more employers working. Thus, our software must be able to handle 30 or more users. Testing will be done to make sure that up to 100 users are handled by the system. This of course also depends on the server features that the software is running on.

1.5 References

The major references used in the preparation of this document are:

1. The IEEE 829-1998 Standard document for Software Test Specification
2. Software Engineering book by Roger S. Pressman
3. Systems Validation & Verification, Quality and Standards course website at The Monash University. See http://www.csse.monash.edu.au/~sitar/CSE4431-MUSE2002/

2 TESTING STRATEGY

There are two types of testing to be used in testing our software system. These two types consist of black-box and white-box testing. Black-box testing is done when testing the software system using only the interface to be used by the end-users. This test is very useful in observing how the end-user will use the system and observing the system from a wider perspective. Black box testing has been conducted right from the beginning of the development after each module development. The SmartSoft team has been testing the SmartGuardian system by surfing on the various web sites on the web, by using the Controller module in inserting rules and policies and in many other situations.

The other type of testing is known as white-box testing. This type of test will also cover the architecture of the system. The internal working of the system will be tested in the function granularity. Unit testing is included in white-box testing as the input-output characteristics of the system are tested in this method. This type of testing has been conducted in a limited extend up till now. This testing will check for right outputs of functions given a designated set of test inputs. A debugger will be very helpful in determining the causes of the bugs when the outputs are not as expected. Also, the integration of the system will be tested using the white box testing strategies. Module interactions will be observed and checked for the requirements in terms of performance and reliability.

2.1 Software Component Testing

Each module of the system will be tested on its own. This is because of the fact that the requirements of each module differ greatly. For example, the Crawler module will be tested for right content classification while the Wall module will be tested for the correct connections to the remote servers.

2.1.1 SG Wall Testing

SG Wall is at the core of our system. This module acts as a server and makes web connections as requested by the clients. It is important for this module to function correctly since if it doesn't than the system will not much use to anyone. For feature tests, the Wall module will be tested against several web sites with many images and loaded content. It will also be tested against crowded web sites. The success of the Wall module will mostly depend on the ability of the Wall module to correctly forward the web sites to the clients.

As we have discussed previously, the Wall module must be able to withstand stress tests that are to simulate about 100 users at the same time. A separate program to simulate these requests is planned to be written, if a simulator already written is not found to be suitable. There are already many stress simulators for web servers; these can be used for our gateway as well. Possible candidates are searched for on the web.

The interoperability of the Wall module is also another issue that has to be taken seriously. There are many browsers around the market today and many versions of these browsers also. The Wall module must be able to work with different types of browsers. Currently we have tested the module to work with Firefox, Konqueror and Mozilla Browsers; MS IE browser is to be tested thoroughly too.

2.1.2 SG Authenticator Testing

The authenticator module is responsible for the control of the users. It will enable users to log in using username and passwords and reject intrusions. Another point is the session expiration property that has to work correctly so that people forgetting to log out will not impair the system and impose a security threat. The timeout for inactivity is to be controlled by setting it in a configuration file.

This module will be tested by using different user and password combinations to test the successful login in the case of correct pairs and the prevention of logging in the

case of incorrect ones. The correct working of the session expiration will also be tested as this is an important part regarding security issues.

2.1.3 Verifier Testing

The Verifier module is responsible for executing the rules that are defined for the users. The testing will be done in the rule granularity and also at the whole policy level. Each rule will be tested to see if the execution is correct and that the expected results have been received. For example, if a time rule is going to be tested, this rule will be put in a policy and tested individually for correctness.

The whole Verifier is to be tested as well. For this purpose, a policy containing many different types of rules will be assigned to a user group and a user from this group will be surfing the web and testing these rules.

2.1.4 SG Controller Testing

The job of the Controller module is to administer the system via a graphical user interface. The usability of the system thus depends greatly on the Controller. A person not knowing any of the internal database used in the system will be able to add rules, policies etc. to the system and manage the server. The Controller module is developed in PHP, we think it is better to test the Controller module using black-box testing. Changes to the system will be made using the Controller module and the required changes will be inspected in the database. While checking for the changes in the database, phpmyadmin will be used as a web interface to easily track the changes. For the usability of the Controller, end-users will be instructed to use this module to manage the server. Easy use is a central point in Controller, continuous improvements to the interface will be made as needed.

The security of the Controller system is important as well, a login mechanism must be able to keep intruders out of the system. This is another point to test the system.

2.1.5 SG Crawler Testing

The Crawler is responsible for classification of the websites. As stated in the Detailed Design Report, the testing of the Crawler will consist of two parts, as the module itself.

The first testing of the Crawler will be done in the training of the Crawler. In this part, 80% of the training data set will be used for training, and the 20% of the data will be used as a test set. After the feeding of the system, the remaining 20% of the websites will be fed to see if the crawler module can successfully classify these sites according to the previous results. If the wrong classifications are in insignificant numbers, than the system will is said to be successful. If this is not the case than a revision of the classification algorithm has to be made. This will not be a major modification, so that it will not result in rewriting of the whole module. Rather, the modifications will be only slight coefficient changes that will be easily fixed.

2.2 Integration Testing

Apart from the testing of each module, an integrated system testing has to be made to ensure that the system behaves as expected. After the module testing, many of the bugs will be eliminated from the system; a final integrated testing will be carried out to test all features of the system. Required interactions of the modules and the general behavior of the system will be observed and required action will be taken to fix problems as needed.

3 TEST SCHEDULE

Test Task	Deadline	Assigned Person(s)
SG Wall	10 May 2006	Akin
Authenticator	05 May 2006	Ozden, Tayfun
Verifier	05 May 2006	Ozden, Tayfun
SG Controller	01 May 2006	Sedat, Hakan
SG Crawler	05 May 2006	Tayfun, Ozden
Integration Testing	10 May 2006	All Persons