Chapter 7.24
A Graphical User Interface (GUI) Testing Methodology

Zafar Singhera
ZAF Consulting, USA

Ellis Horowitz
University of Southern California, USA

Abad Shah
R & D Center of Computer Science, Pakistan

ABSTRACT

Software testing in general and graphical user interface (GUI) testing in particular is one of the major challenges in the lifecycle of any software system. GUI testing is inherently more difficult than the traditional and command-line interface testing. Some of the factors that make GUI testing different from the traditional software testing and significantly more difficult are: a large number of objects, different look and feel of objects, many parameters associated with each object, progressive disclosure, complex inputs from multiple sources, and graphical outputs. The existing testing techniques for the creation and management of test suites need to be adapted/enhanced for GUIs, and new testing techniques are desired to make the creation and management of test suites more efficient and effective. In this article, a methodology is proposed to create test suites for a GUI. The proposed methodology organizes the testing activity into various levels. The tests created at a particular level can be reused at higher levels. This methodology extends the notion of modularity and reusability to the testing phase. The organization and management of the created test suites resembles closely to the structure of the GUI under test.

INTRODUCTION

Graphical user interfaces (GUl) are an important part of any end-user software application today and can consume significant design, development, and testing activities. As much as half

of the source code of a typical user-interaction intensive application can be related to user interfaces (Harold, Gupta, & Soffa, 1993; Horowitz & Singhera, 1993). GUIs provide an easier way of using various functions of the application by organizing them in a hierarchy of options and presenting only the options which make sense in the current working context. GUIs help users concentrate on the problem instead of putting efforts in remembering all the options provided by the software application that is being used to solve the problem, or searching for the right option from a huge list of options provided by the application. Graphical user interfaces organize the standard user actions and working paradigms into various components that are presented graphically to the user during various usage and application contexts. GUIs enhance the usability of an application significantly. However it also makes application development, testing and maintenance significantly more difficult (Myers, 1993; Wittel & Lewis,1991). The nature of GUI applications, their asynchronous mode of operation, nontraditional input and output, and hierarchical structure for user interaction make their testing significantly different and difficult from the traditional software testing.

Functional and regression testing of graphical user interfaces is significantly more complex than testing of traditional non-GUI applications because of the additional complexities mentioned in the previous paragraph. A number of commercial tools, like Mercury Interactive's WinRunner, XRunner and Segue Software's SilkPerformer, are used in the industry to test graphical user interfaces. These tools provide capture/replay capabilities to test a graphical user interface. Although functionality provided by these tools is sufficient for typical recored/replay scenarios but they lack an underlying model that can provide more information about the test coverage or to determine the quality of the user interface from a particular functional or implementation perspective. These tools also do not provide a framework that assists in organized and modular testing. The methodology presented in this article uses user interface graphs (UIG) as a framework for organization of test scripts, generation of modular test suites, and coverage analysis of a test execution.

In this article, we propose a methodology for regression testing of graphical user interfaces, with and without a formal specification of the application under test. The remainder of this article is organized as follows: Section 2 highlights some of the best practices and recommendations that help in testing a GUI application in an organized fashion, improve efficiency and effectiveness of testing, reduces possibility of errors, and minimizes repeated work. Section 3 describes the major steps of the proposed methodology. It also introduces a sample X application, called Xman, which is used to demonstrate the effectiveness of the suggested strategy. Section 4 demonstrates the testing methodology when formal specifications of the application under test are not available. Section 5 illustrates the proposed testing methodology when the formal specifications of the application under test are available. This section also describes the way statistics are collected during a testing activity and how those can be used to improve the quality of the testing. Section 6 points out the situations when a modification to the application under test might require tuning or recapturing of some of the test scripts. Section 7 concludes the article by summarizing our contribution and providing hints about the future related work.

GUI TESTING: BEST PRACTICES AND RECOMMENDATIONS

In this section, we highlight some of the sought features, well-knows best practices and recommendations for planning a testing activity for a graphical user interface.