Chapter 87
How Much Automation can be done in Testing?

Izzat Alsmadi
Yarmouk University, Jordan

ABSTRACT

It is widely acknowledged that software testing stage is a stage in the software project that is time and resources' consuming. In addition, this stage comes late in the project, usually at the time where pressure of delivery is high. Those are some reasons why major research projects in testing focus on methods to automate one or more of activities in this stage. In this chapter, description of all sub stages in software testing is explained along with possible methods to automate activities in this sub stage. The focus in this chapter is on the user interface of the software as it is one of the major components that receives a large percentage of testing. A question always raised in testing is whether full test automation is possible and if that can be feasible, possible and applicable. While 100% test automation is theoretic and impractical, given all types of activities that may occur in testing, it is hoped that a maximum coverage in test automation will be visible soon.

INTRODUCTION

Part of software project development, it is very important to make sure, before product delivery, that the software is working as expected without serious flaws or problems. Testing the software, like any other product, can be seen from several perspectives. It can be seen based on what we are testing. For example, each deliverable in the software project (e.g. the requirements, design, code, documents, user interface, etc) should be tested. Moreover, in the code, which is the main deliverable, we may test the code based on the user and functional requirements or specification (i.e. black box testing). In this level, we are testing the code as a black box to make sure that all services expected from the program are: existed, working as expected and with no problem. On this perspective, we may need also to test the internal structure of the code (i.e. white box testing) based on the code

DOI: 10.4018/978-1-4666-4301-7.ch087

structure or any other element. Testing can be also divided based on the sub stages or activities that occur in testing. For example, the testing stage includes the main activities: test case generation and design, test case execution and verification, building the testing database or oracle, coverage evaluation and assessment, results reporting and so on. Software testing tries to ensure that software products are not developed heuristically and optimistically. This software engineering stage ensures that the developed software is, ultimately and hopefully, error free. However, in reality, no process can guarantee that the developed software is 100% error free). In all cases, it is widely agreed that, this stage takes a large percent of the overall project resources.

Software test automation is the process of achieving one or more of software testing activities mentioned earlier programmatically or automatically with no user intervention (or with minimal user intervention). Software test automation is expected to cost more at the beginning while save more eventually. There are extra costs for test automation framework at start-up. However, once correctly implemented, it is widely accepted that test automation will decrease the cost as it reduces the required number of human testers and hence reduces the cost of testing. There are several repetitive tasks in testing that take a long time and that occur very often (e.g. regression testing, reporting, test cases' execution, etc.). For such activities, test automation is the best choice.

As explained at the beginning of this chapter, the focus of this chapter is in test automation for Graphical User Interfaces (GUIs). GUIs are getting more and more focus and roles in new applications. Earlier, the GUI has less important role and many program events can be executed with very simple Console commands. Developing rich GUIs that can handle complex types of application-user interactions became a trend in all new applications. A typical programming Integrated Development Environment (IDE) such as.NET, Eclipse or NetBeans includes lots of types of GUI components. Every time a new release is developed of those IDEs, one of the expected additions is new or richer GUI components. Such richness in GUI components add extra overhead to software testing where testing such components extensively require several different complex scenarios.

Challenges in GUI Test Automation

GUI test automation is a major challenge for test automation activities. Most of the current GUI test automation tools are partially automated and require the involvement of users or testers in several stages. Examples of some of the challenges that face GUI test automation is: dealing with continuously new controls or components, the dynamic behavior of the user interface, timing and synchronization problems, etc.

Test automation tools are still complex and expensive. They don't fully replace testers. They can be usually used to re-execute those repeated tasks. Companies consider taking the decision to buy, or not, a GUI test automation tool as a tough decision since they don't know how much time it will require upfront for setup. They also don't know how much of test automation tasks can be fully or partially automated. Software testers all agree that complete coverage in test automation is impractical as we also agree that complete coverage in manual testing is impractical too.

In record/play back test automation tools, we have to retest the application in case of any change in the functionalities or the GUI of the program. In many cases, this is seen as making such tools impractical especially as the program GUI is expected to be continuously evolved and changed.

GUI Test Automation Tools

There are several GUI test automation tools available in the market. Some of the larger vendors for such tools include: IBM Rational, Mercury, and Segue. Trying to build a GUI test automation tool is like trying to make a new operating system in