COMPARATIVE STUDY OF OPEN SOURCE CAPTURE AND REPLAY TOOLS

Sumangali K, Nilesh Ghatol
Department of Information Technology, School of Information Technology and Engineering, VIT University, Vellore – 632 014, Tamilnadu, India.
Email: ksumangali@vit.ac.in

Received on 25-10-2016 Accepted on 02-11-2016

Abstract

Graphical user interface (GUI) has become extremely vital because of its pervasive use in computers and software products. However, testing the GUI still continue to be a big challenge to ensure the quality of the system and to enhance user satisfaction. Testing something manually is acceptable only by ad hoc systems; however it often results in large amounts of manual labor, thus increasing the cost. Using GUI automation tools to support the testing process helps to overcome the problems associated with manual testing. In this paper we compare four open source GUI automation tools in terms of ease of use, and GUI automation characteristics. We compare the tools Sikuli, AutoIt, Jubula and Marathon considering generic and GUI automation characteristics. After evaluating each tool, we choose the one that showed the best result. The conclusion of this study will serve as a guide for beginner testers or companies that consider employing open source GUI automation tools to automate the GUI testing process.

Keywords: Sikuli, AutoIt, Jubula, Marathon.

1. Introduction

Nowadays, most software products have a graphical user interface or GUI in short. The GUIs have an important role in human-computer interaction, since they make it easier for people with less computer skills to use the learning system. The user interacts with software through several GUI elements called widgets such as Windows, buttons, menus, text fields, etc. These will help the user to accomplish its objectives in using the software. Testing the GUI is the process of testing the graphical user interface of a software product. This is important because often the GUI is the only part with which the user interacts, ensuring the quality of the system and the user's satisfaction. However, it not only involves testing the interface, but also the functionality of the software. It is expected that it disclose many of the test bugs in the
software with the bugs not being actually in the GUI. Therefore, if the test is performed at the beginning of the development process, it can increase the efficiency of the application, decreasing risk and increasing the quality.

However, there are many different issues that arise when it comes to the GUI tests. Some of them are the problems in the generation of test cases, the large input space, and non-usefulness of the test cases in regression tests. There are summarized problems associated with the testing of GUIs and are referred to the approaches of capture and playback as key time spenders. It is also costly compared to the manual testing process, since it requires a high knowledge of the functioning of the test tools. Some authors refer to the costs of learning as hidden costs.

To validate the proper operation of the GUI-based applications, GUI automation tools are commonly used in acceptance or functional tests. The basic concept of these tools is that the tester performs the target application and creates the test cases manually (predefined sequence of actions) using these tools. GUI automation tool captures user actions in a log file. The tool uses this test script to playback all actions recorded later without the need of a human user. Therefore, GUI automation tools support automatic regression tests by repeating the test scripts on a modified version of the target application. The benefits of using automation tools are that they save the time to encode the test scripts manually and also make it easier for testers, who do not have great programming knowledge, automate the test cases. The test cases that were recorded can be repeated as required, several times. Nevertheless, these tools also have some drawbacks. First, the GUI automation tools do not support the automatic generation of test cases. Tester has to define the test cases based on their domain knowledge and experience, and then run the tests manually so that the tool can capture the events happening. The test scripts that are recorded are sometimes inefficient and require manual debugging or intervention. It should be understood that the automation tools only know how to interact with widgets since they are unaware of the application that is being operated. And considering the regression tests, test cases generated by the automation tool often require a lot of manual work to modify them, since any change in the GUI can cause a break in a test case.

This paper compares four open source tools to automate GUI. Our motivation to choose compare open source tools was caused because of two major reasons. Any small or medium-sized enterprise does not need to spend money in the acquisition of software to automate the GUI testing process, and the second is that we want to assess what is the best tool to automate GUI. Hence, our study provides the importance for small and medium-sized enterprises and used by testers with little experience.
We organize the paper as follows. In section 2, we provide some previous works on this idea which is used as a starting point. In section 3, our contribution is to compare tools in terms of user experience, which involves among other things, learning and ease of use, which is not covered in the study of other authors. Next, in section 4, we conclude this article with most preferred tools in GUI environments.

2. Open source tools to capture and Playback

This section presents the four open source tools that automated GUI and were compared: AutoIt, Sikuli, Jubula and Marathon. These tools were chosen on the basis of previous works and also based on the popularity of the tools, and the number of downloads and user rating on different repository where the tools are distributed. The analysis resulted in the choice of these four tools with the highest number of downloads and better acceptance by the users. It is noteworthy to mention that all the tools are multiplatform accepted. In the following subsections, we present the main features of each tool on the basis of the information provided on its Web sites.

A. Sikuli

Sikuli is a graphical user interfaces (GUI) automation tool which uses “Visual Image Match” method. Every element should be captured as an image and stored inside the project. Sikuli will initiate GUI interactions based on the image visual match which we have passed as the parameter along with all methods. Sikuli can be much useful to automate flash objects which do not have ID or name. It is useful in the situations where a stable GUI is available. Window based applications can also be automated using Sikuli tool. It is also possible to automate Flash Games, Adobe Video or Audio player on website using this tool. It makes coding easier, with simple API. Sikuli currently uses Python as the scripting language.

Some of the basic characteristics of Sikuli are as follows:

1. It can be used to automate Flash Objects or Flash Websites.
2. It can be useful to automate Window based application.
3. It provides simple API. All methods can be accessed using screen class object.
4. It can be easily integrated with Selenium and all other tools.
5. We can automate desktop applications.
6. It uses powerful “Visual Match” mechanism to automate desktop & flash objects.
Therefore, Sikuli automates anything you see on the screen. It basically uses image recognition to identify and control the GUI components. Sikuli is useful in situations when there is no easy access to a GUI's internal or source code.

**B. Marathon**

Marathon Integrated Testing Environment or Marathon ITE, is an easy-to-use, cross-platform and affordable Java/Swing™ GUI Test automation framework. Marathon ITE's inbuilt script recorder can be used to create readable, clean test scripts either in Ruby or Python. Advanced features like create-data driven-tests, extract-method refactoring and object map editing allows to create resilient and maintainable test suites. Some of its features are:

1. Marathon consists of a recorder that allows to easily creating test scripts.
2. There is no need to learn a specific language to develop Marathon ITE scripts.
3. It provides multiple mechanisms of object naming and recognition. It is handled by Naming Strategy – a pluggable interface.
4. It exploratory testing mode provides recording of test session in the background and also allows you to capture screenshots and annotate them for noting the findings.
5. It facilitates Semi Automated tests using checklists. During recording of a test case, at some point, one can insert a checklist.
6. It provides Convert to Module Method refactoring which is handy for creating such reusable script-lets.
7. It provides APIs so that additional component resolvers can be developed for those cases where it is needed.
8. It provides you with flexibility to design your test projects as you seem fit.

Thus, Marathon is a testing framework for GUI applications developed using Java/Swing. It is used to reduce the effort required for creating resilient test suites.

**C. Jubula**

Jubula gives robotized utilitarian GUI testing to different sorts of utilizations. It is gone for groups who need their robotized tests to be composed by test specialists from the client viewpoint, without obliging any coding exertion. Jubula tests consolidate best practices from programming improvement to guarantee long haul practicality of the computerized tests. Jubula will likewise attempt to be a stay point for a more extensive testing degree including prerequisites investigation, code scope of Java applications and test measurements.
Jubula was contributed by BREDEX GmbH from their past business device GUIdancer. BREDEX GmbH likewise offer a Jubula standalone that consolidates the Jubula center and offers extra highlights and choices for profitability and solace in testing. This highlight contains the Jubula center highlights, online documentation, an inserted AUT Agent, and offers the Swing, SWT/RCP/GEF and JavaFX tool stash. Jubula tests control the Application under Test (AUT) by means of the GUI so that the tests reflect precisely the same viewpoint that a client has when working with the product. A long way from being the fragile and flimsy robotization method it is regularly asserted to be, test mechanization through the GUI can without much of a stretch stay aware of advancement if best practices, for example, reuse and coherence are stuck to. Hence, Jubula offers an option way to deal with the customary catch and replay strategies. Contrary to recording activities in the GUI, tests are made progressively from reusable modules.

The steps necessary to make tests flexible, robust and intelligent are transparent and easy to implement from the user’s point of view. As a result, automated tests that can accompany a project throughout its whole lifecycle are obtained.

D. AutoIt

AutoIt v3 is a freeware BASIC-like scripting dialect intended for computerizing the Windows GUI and general scripting. This permits the user and its clients to collaborate with the user-written scripts. It can be as expound or straightforward as the user likes, containing numerous controls, or simply a pair.

Main features of AutoIt:

1. Interact with all standard windows controls
2. Scripts can be compiled into standalone executable
3. Create Graphical User Interfaces (GUIs)
4. COM support
5. Regular expressions
6. Directly call external DLL and Windows API functions
7. Scriptable RunAs functions
8. Detailed help file and large community-based support forums
10. Unicode and x64 support
3. Analysis and Discussion

This section presents our approach to comparison of GUI automation tools introduced in the previous section. The experimental environment where the tools were installed and run consisted of a Dell portable computer with processor features Intel Core i3 (2.40GHz); 4 GB of RAM and system operating Windows 7 (64-bit as well as 32-bit) with Java 1.7.

Criteria used for comparison were divided into two major groups: General Features - which aim to compare some basic features which are very important for user satisfaction, and the GUI automation - which can be understood by its name, automates the GUI. The comparing features of the tools and the analysis of the results are presented in detail in the following subsections. We are comparing the tools based on their features, which is provided for GUI test. Below table shows the features of each tool, on that basis comparison was done.

Table 1: Features of Tools.

<table>
<thead>
<tr>
<th></th>
<th>Sikuli</th>
<th>Marathon</th>
<th>Jubula</th>
<th>AutoIt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automate Flash Objects or Flash Websites</td>
<td>No need to learn a specific language to develop MarathonITE scripts</td>
<td>No recording</td>
<td>Easy to learn BASIC-like syntax</td>
<td></td>
</tr>
<tr>
<td>Window based application</td>
<td>Provides multiple mechanisms of object recognition</td>
<td>No program code</td>
<td>Manipulate windows and processes</td>
<td></td>
</tr>
<tr>
<td>Simple API accessed using screen class object</td>
<td>Consists of a recorder that allows to easily creating test scripts</td>
<td>All layers tested</td>
<td>Digitally signed</td>
<td></td>
</tr>
<tr>
<td>Easily integrated with Selenium and all other tools</td>
<td>Flexibility to design</td>
<td>Offers the Swing, SWT/RCP/GEF and JavaFX</td>
<td>Compatible with Windows 95 / 98 / ME / NT4 / 2000 /</td>
<td></td>
</tr>
</tbody>
</table>
Desktop applications | Additional component resolvers can be developed | Embedded AUT Agent | XP / 2003 / Vista / 2008

“Visual Match” mechanism | provides Convert to Module Method refactoring | Online documentation | Scriptable RunAs functions

4. Conclusion

After evaluating the four open source GUI automation tools, we conclude that Sikuli is by far the best tool among them. Sikuli can be used by the novice testers or the employees who have poor or no knowledge of programming skills in a company. It is easily operated and has a user-friendly graphic interface.

References


Corresponding Author:

Sumangali K*,

Email: ksumangali@vit.ac.in