Extension of Sikuli Tool to Support Automated Tests to Windows Phone Context-Aware Applications

Elizângela Santos da Costa
Product Validation
Institute of Technology Development
Manaus-AM, Brazil
email: elizangela.costa@indt.org.br

Rodrigo dos Anjos Cruz Reis
Institute of Computing
Federal University of Amazonas
Manaus-AM, Brazil
email: rdgdosanjos@gmail.com

Arilo Claudio Dias-Neto
Institute of Computing
Federal University of Amazonas
Manaus-AM, Brazil
email: arilo@icomp.ufam.edu.br

***Abstract* — The wide diffusion in the use of mobile devices has brought the need to improve the process of verification and validation in mobile applications. The usual way of interaction between these apps and users is through device interface. For context-aware mobile applications, the number of interactions that a user can perform is much larger if compared to common applications. Thus, manual testing execution turns out to be an exhaustive and error-prone activity. The main contribution of this work is to propose the creation of new functions in Sikuli tool to automate tests for context-aware mobile application developed to the Windows Phone platform in order to develop reliable applications using an effective test strategy.**

Keywords-Context-awareness; Automated Testing; Sikuli; Mobile Testing.

I. INTRODUCTION

On mobile platforms, the main form of interaction between users and applications is through Graphical User Interface (GUI). Thus, since mobile applications development is increasing greatly, GUI becomes more complex and more concern is required for its quality.

A method to evaluate the quality of software through its GUI is by performing a testing technique called GUI Test [1]. This type of testing must simulate the sequence of events performed by users. The large number of input possibilities for this sequence makes the GUI test a complex activity and requires a lot of manual effort for the testing process.

An important factor to be considered during the evaluation process of mobile applications is the context-awareness characteristic [2]. This characteristic aims to describe different contexts in which applications are subjects, meaning that they can react differently to changes in their environments. Different contexts in different applications are more likely to generate failures and the criteria of coverage to reach all possibilities should be proposed. An alternative to reduce the effort in these tests is the adoption of automated testing.

The rest of this paper is organized as follows. Section II describes the background. Section III describes the tool extended in this work. Section IV addresses the proof of concept performed with the extended tool. Finally, conclusion and future work are described in Section V.

II. BACKGROUND

Ubiquitous Computing is defined as: "an area of research that studies the integration of technology to human activities in a transparent way, when and where needed" [3]. Context-awareness is a sub domain of Ubiquitous Computing. *Context* is defined as any information that can be used to characterize the situation of an entity [2]. Devices, services and software components should be aware of their contexts and automatically adapt to your changes, characterizing the *context-awareness* [4].

A concept for *mobile application testing* and *GUI test* can be found in [5][6] respectively. *Sikuli* is a tool that uses visual approach to search and automate GUI tests using screenshots. Through this tool, testers can write visual scripts that specify the components to interact and what visual feedback to expect, it has the advantage of being independent of any platform [7].

III. EXTENDING THE SIKULI TOOL

The work was developed in four steps. First, the context elements of the device defined to be automated with respect to screen orientation, phone battery level, internet connection and location.

Changes in context elements were chosen in the second step. Thus, screen orientation can be positioned vertically or horizontally, battery has several levels represented by a percentage, connection can be enabled or disabled for both Wi-Fi and 3G or 4G connections, location can be enabled or disabled. Figure 1 resumes the structure of development. In the third step, the following automation scripts in Sikuli were developed:

- ***UtilWP***: Main functions for test automation;
- ***PC***: Script for proof of concept, it calls the functions created in UtilWP script;
- ***NivelBateria*** *and* ***NivelBateriaH:*** They store a database for existing images of battery levels in portrait and landscape;
- ***Scroll***: Contains functions that perform the sliding movement;
- ***Alfabeto***: It stores a database for alphabet letters helping in function of open app.

 ISBN: 978-1-61208-418-3

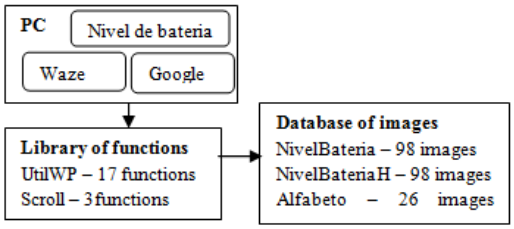

Figure 1. Structure of scripts developed.

It used the Windows Phone emulator called *Project My Screen App* [8] to project the application screen to be possible run the scripts. This is the last step.

IV. PROOF OF CONCEPT

To analyze the functions created, three mobile apps were chosen as shown on Table 1. To validate the reaction of mobile applications to the contexts, first it was analyzed how they should behave in cases of certain changes in device contexts.

TABLE I. CONTEXT VARIATION IN THE MOBILE APPS PER ELEMENTS

App	Screen Orientation	Internet Connection	Location	Battery
Google	X	X		
Waze			X	
Nível de Bateria (Battery Level)				X

After that, the execution of scripts was implemented using the functions created, as described in Section III. An example of automation can be seen in Figure 2 that validates the behavior of Waze app. In short, the script defines the expected behavior through image (line 2 and 3) for both location ON and OFF, modifies the status of the location property of the phone (to ON), opens the app and verifies whether Waze has adapted to the new context or not.

```
def WazeTest():
    locOn= 
    locOff= 
    switchApp("Project My Screen App")
    location=LocationOn()
    if (location=='on'):
        OpenApp('w',  )
        print '--Location connected--'
        wait(locOn,5)
        AssertEqual(locOn)
    else:
        print 'Connect location failed'
```

Figure 2. Test case for Waze.

The final steps were to run the scripts for the remainder of the apps and compare the results after execution.

As demonstrated, a tester is able to test a context-aware application once the right screen is defined to each state of context element. The additional libraries provide an easy way to write automated tests because the changes in device were automated before such as basic functions like open a mobile app. Now that changes in location are automated in the extended library, any mobile app that is context-aware to location can be tested with less effort, not only Waze.

In order to avoid errors for whom uses the tool, some care has been taken, such as treatments for occasional exceptions images that are not found. The device style pattern should also be noticed, since it is possible to change themes. Inserting wait commands with the pictures expected before checking equality between screens helps in fault occurrence prevention. Check if the image inserted in the script will be recognized by matching preview (Sikuli property) provides the preliminary recognition of faults that can be generated.

V. CONCLUSION AND FUTURE WORK

To provide a better process on validation methodology, it was proposed to use an existing tool for automating tests, Sikuli, extending its functions to create new ones that reduce manual efforts for the testing activity. Improvements of the implemented functions can be achieved by creating an image library for common buttons making the scripts cleaner and easier to maintain.

Some limitations were found throughout this work, and the most critically noted ones are instability in the Sikuli tool and operating system restriction, since Windows must be 8 or greater, to support the functionality of the control device by emulator.

Despite the existing limitations, the support offered to reduce the manual testing tasks and proved that it is feasible to automate context-aware mobile applications observing its adaptations to changes in addition to offer reduction of testing execution time and manual effort.

REFERENCES

[1] A. Ruiz and Y. W. Price, “GUI Testing made easy,”. Testing: Academic & Industrial Conference - Practice and Research Techniques. IEEE 2008, pp. 99-103, doi: 10.1109/TAIC.PART.2008.11.

[2] D. Amalfitano, A. Fasolino, P. Tramontana, and N. Amatucci, “Considering context events in event-based testing of mobile applications”, IEEE Sixty International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Luxemburgo, Mar. 2013, pp. 126-133.

[3] M. Weiser, “The Computer for the 21st Century”. Mobile Computing and Communications Review – Special Issue Dedicated to Mark Weiser, vol. 3, no. 3, July 1999, pp. 3-11.

[4] J. L. B. Lopes, “Exehda-on: an approach based on ontologies for context-awareness in pervasive computing”, Dissertation (Master in Computer Science) – School of Informatics, Catholic University of Pelotas, 2008, p. 198.

[5] J. Gao, X. Bai, W. Tsai, and T. Uehara, “Mobile Application Testing: A Tutorial”, In:,” Computer, vol. 47, no. 2, Feb. 2014, pp. 46-55.

[6] A. M. Memon, M. E. Pollac, and M. L. Soffa, “Hierarchical GUI Test Case Generation Using Automated Planning,” IEEE Transactions on Software Engineering, vol. 27, no. 2, Feb 2001, pp. 144-155.

[7] T. H. Chang, T. Yeh, and R. Miller, “GUI Testing using Computer Vision,”. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, USA, 2010, pp. 1535-1544.

[8] Project my phone screen to a TV or PC. Available from: http://www.windowsphone.com/en-us/how-to/wp8/connectivity/project-my-phone-screen/2015.05.30

 ISBN: 978-1-61208-418-3