Automated Stress Testing of Windows Mobile GUI Applications

Nizam Abdallah, Sita Ramakrishnan
Faculty of Information Technology
Monash University
Melbourne, Australia
e-mail: {nizam.abdallah, sita.ramakrishnan}@infotech.monash.edu.au

***Abstract*— Mobile devices are now a common tool for personal, business, entertainment and communication use. The increase in usage of Smartphone devices has led to the increase in development of mobile software applications. The testing of mobile applications is made difficult as there are many different types of devices available that may be targeted by the same application. In addition to this, mobile devices are a resource constrained device, with less memory and processing power than the desktop PC. However, even with these constrained resources, performance and robustness are still an important factor to consider when developing mobile software applications. This research paper discusses the current status and proposed work of a research project involving the use of artificial intelligence, fuzz-testing and automated GUI testing techniques for Windows Mobile devices. It describes the current development status of an automated GUI stress testing tool, *Torqueo*. Then, it details the various types of scenarios for which this tool has been utilized and also outlines the advantages and limitations of this research study.**

Keywords-Windows Mobile; GUI Applications; .NET Compact Framework; Automated Testing; Stress Testing

I. INTRODUCTION

More and more people are relying on their mobile devices, not only for voice and SMS communications, but also for, business, email, scheduling, social networking, playing games and listening to music. The power of these mobile devices has increased over the years and so too has the software that is available for these devices. Automated GUI testing has proven to be a valuable part of the software development life-cycle. Previous work has demonstrated that if implemented correctly, automated GUI testing is an effective method for improving the quality of software and makes the software testing process more efficient and effective [5]. This study looks at the development of a Windows Mobile automated GUI stress testing tool, *Torqueo,* which attempts to bring the benefits of test automation to mobile software development.

II. BACKGROUND

The popularity of mobile devices has increased significantly over the last few years. According to the CIA World Fact book in 2005 the total number of cellular subscribers worldwide was 2,168,433,600 [2]. Currently there are a large number of mobile device manufacturers including but not limited to Nokia, Palm, Sony, Motorola, RIM, Samsung, Apple and HTC. The devices support one of the many mobile platforms including Symbian, PalmOS, Windows Mobile, Windows CE, Android, Blackberry, Apple and Linux.

This increase in popularity has led to an increase in the number of mobile applications developed for these devices. As a result the software developed for mobile devices needs to be of high quality, not only in terms of functionality but also in terms of reliability, robustness and stability. Applications need to be able to operate for long hours and not interfere with other device functions, or degrade performance of the device.

Microsoft's Windows Mobile platform was developed to run on various hardware including mobile phones, PDAs and other electronic devices such as GPS navigation systems and portable entertainment units. Windows mobile is currently released in three editions, including:

- Windows Mobile Standard
- Windows Mobile Professional
- Windows Mobile Classic

All of Microsoft's Windows Mobile platforms are built on Microsoft Windows CE Operating System.

Traditionally automated GUI testing tools allow testers and developers to record user actions into scripts and play these scripts back performing the exact same actions as those that were recorded [5]. These recorded scripts work well as long as the GUI does not change. If the GUI or GUI logic is modified, the scripts will fail and generally need to be recorded again, leading to high maintenance costs [5]. Recent research has demonstrated methods for optimizing automated test case selection based on artificial intelligence techniques, using planning [7], [8], [9] and neural networks [13]. Memon, Pollack and Soffa [8], present a GUI test case selection technique using planning. In this planning approach, a GUI application is analyzed and a set of *initial* and *goal* states are created, then using this state information, the use of a hierarchical planner is employed to create test cases. Ye, Feng, Lin and Zhu [13], have also researched the area of Artificial Neural Networks for test case selection. In their research, a trained Neural Network is used to minimize the number of GUI test cases selected for execution. The test cases selected are those that have the most chance of finding a defect within the application under test. While these research projects were successful, they were aimed at GUI applications running on desktop personal computers. This approach may result in a great performance impact if implemented on a mobile device. Furthermore, Automation frameworks for mobile applications have also been researched [1], [11] but these involved the user to create tests for each application, then to execute these tests on the target device, leading to the same problem mentioned above, that if the GUI or GUI logic is modified the tests will need to be re-created.

This research study includes the development of an automation framework for stress testing Windows Mobile GUI applications. The aim of this tool is to automatically generate tests for .NET Compact Framework Applications and automatically execute these tests. Similar stress testing tools have already been developed including Microsoft's Hopper Test Tool. The Microsoft Developer Network (MSDN) documentation [10], describes "Hopper is a software test tool that simulates input stress on Windows Mobile powered devices. Hopper will stress all applications that are available through the menu system by rapidly sending keystrokes and screen taps in a random fashion. By sending a large number of user inputs very rapidly, Hopper can quickly isolate troublesome scenarios and find bugs in your applications". While Hopper can find many issues, generally these issues are hard to reproduce as the inputs performed by Hopper are random and not recorded. However, the MSDN documentation [10], explains that "Sometimes, bugs found by Hopper are very difficult to find and fix, but their value should not be underestimated. Hopper will help you find system and application scenarios that you may not have thought of." Microsoft's Hopper can be customized. Microsoft explains that the tester or developer will need to make changes to the *FocusApp* source code, which is included as part of the Hopper tool. *FocusApp* allows the tester to specify the name of the executable for the application under test as well as a sleep timeout by modifying the *FocusApp* source code. Hopper will then check to ensure that the application under test is in focus. The frequency of this check depends of the value of the sleep timeout. A shorter timeout value will result in a more frequent check. This is an important factor to consider as the application under test may lose focus due to the random nature of the input provided by Hopper.

Motorola have also researched this area of automated stress testing and have developed iHopper [3] and iRobot [4] which perform similar functions to Microsoft's Hopper test tool. However, Motorola have added the ability to record and playback the actions performed. Furthermore in Motorola's iHopper the probability of certain input varies depending on the key-press to be simulated [3]. The limitations of iHopper is that it is designed to run only on the iDEN^{TM} UIS phone, while the iRobot tool is will only run on the iDEN^{TM} Smartphone.

III. Proposed Work

The automated GUI stress test tool developed in this research study is named *Torqueo*. *Torqueo* is the Latin translation for test, twist, torture or spin. Currently, *Torqueo* includes not only the functionality similar to Microsoft's Hopper, Motorola's iHopper and iRobot, but also the ability to run on any Windows Mobile device running the .NET Compact Framework. The tool will also be extended to include GUI object recognition and automated test case generation and execution using artificial intelligence techniques.

The main aims of this research project are:

•Run on any Windows Mobile device with the .NET Compact Framework installed

•Minimize the performance impact of running the tests on a mobile device

•Minimize the amount of coding required to create tests

•Ability to automatically map GUI objects in the application under test

•Automatically generate tests cases

•Automatically execute test cases

Basic Fuzzing techniques are currently being used in *Torqueo*. Sutton, Greene and Amini [12], describe Fuzzing "as a method for discovering faults in software by providing unexpected input and monitoring for exceptions." Initially, random input is being generated in the form of stylus taps and button presses; however this randomness will be refined once GUI object recognition is added to *Torqueo*.

GUI object recognition is being added to *Torqueo* to allow the tool to recognize .NET Compact Framework GUI controls using .NET Reflection. .NET Reflection is a feature in both the .NET Framework and .NET Compact Framework. Gilani et al [6] describe that ".NET reflection allows an application to search through and examine assembly metadata programmatically at runtime. Once the application has used reflection and metadata it can instantiate classes, invoke class methods and modify their properties". Once *Torqueo* can recognize these GUI objects via .NET Reflection, tests can be automatically generated for each type of GUI object and execution can be performed using artificial intelligence techniques. Various artificial intelligence techniques are being considered, and previous works [7], [8], [9], and [13] have already demonstrated a successful approach to using intelligence methods in automated GUI testing, including, test case generation, test case selection and test execution.

Furthermore, development is underway to make *Torqueo* a lightweight agent running on a device connected to a desktop PC. Not only will this enable the agent to run on multiple Windows Mobile devices simultaneously, it will also allow for tests to be generated, executed and results reported centrally. Furthermore, this agent based architecture will also remove the overhead and performance impact on the device itself.

IV. TORQUEO

Currently, *Torqueo* is a stand-alone device based automated GUI stress testing tool that can not only generate random input in the form of stylus taps and hardware button presses but can also can also execute pre-recorded XML test scripts and perform common device functions such as file IO, voice and data communications and log performance status. A list of current features is shown in Table 1 below.

Table 1. Torqueo Features

Feature	Description
Record and playback	Play back the pre-recorded actions
Random Input	Generate random stylus and key press events
Scripted Input	Execute XML based tests
Phone functionality	Make phone calls and send SMS
Data connections	Initiate a data connection/browse internet
Database IO	Create/Read/Write to SQL Server CE database
File IO	Read/Write to a file
Performance tests	Log CPU and Memory Usage

The only requirements for *Torqueo* are that the .NET Compact Framework 2.0 or later needs to be installed on the device. To make use of the database functions SQL Server CE 3.0 also needs to be installed on the device. Table 2 shows the different devices on which *Torqueo* has been used.

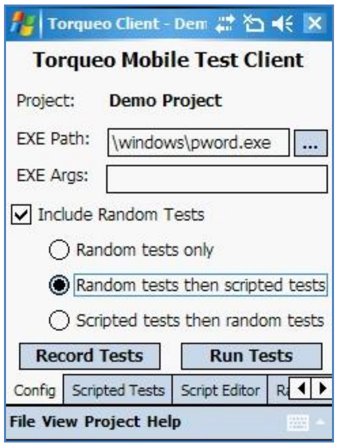
Table 2. Tested Platforms

Manufacturer	Device Model	Microsoft Platform
Dell	X50	Pocket PC 2003
Casio	E-115	Pocket PC 2000
Motorola	MC35	Windows Mobile 5.0 Professional
Samsung	i320N	Windows Mobile 5.0 Smartphone
Dopod	s301	Windows Mobile 5.0 Smartphone
HP	iPaq rx4500	Windows Mobile 5.0 Classic
iCOP	eBox-2300	Windows CE 6.0
Motorola	MC75	Windows Mobile 6.1 Professional
Motorola	MC70	Windows Mobile 5.0 Professional
Motorola	VC6090	Windows Mobile 6.1 Professional
Intermec	761	Pocket PC 2003
Intermec	CN3	Windows Mobile 6.1 Professional
Motorola	Q9	Windows Mobile 6.0 Smartphone

Torqueo can perform tests in 3 ways:

1. Generate random input
2. Execute scripted XML tests
3. Perform both random and scripted tests

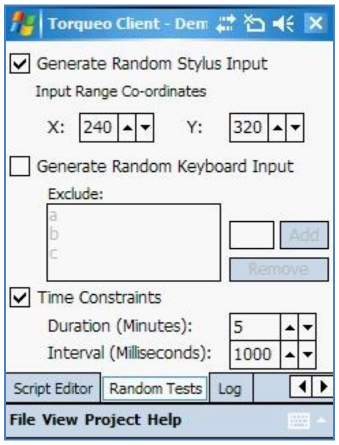
Fig. 1 below shows the main *Torqueo* screen, where the test analyst can select the application to be tested, and the types of tests to be executed.



Project: Demo Project
EXE Path: \windows\pword.exe
EXE Args:
☑ Include Random Tests
○ Random tests only
● Random tests then scripted tests
○ Scripted tests then random tests

Figure 1. Torqueo Main Screen

While *Torqueo* can generate random input, constraints can be applied to the random input data generated. Fig. 2 shows how the test analyst can configure screen co-ordinates for the random stylus taps, also exclude certain hardware buttons from being simulated if required and finally the duration of the random test can be specified along with the interval between each event.



☑ Generate Random Stylus Input
Input Range Co-ordinates
X: 240 Y: 320
☐ Generate Random Keyboard Input
Exclude:
a
b
c
☑ Time Constraints
Duration (Minutes): 5
Interval (Milliseconds): 1000

Figure 2. Torqueo Random Test Configuration

In addition to randomly generated input, the test analyst can create XML test scripts that perform pre-determined actions. Fig. 3 shows a sample XML test script while Table 3 shows the list of actions currently supported by the XML script.

```
<testsuite name="PocketWord">
  <test name="Hello World">
    <clickMouse x="20" y="305"/>
    <sleep time="5000"/>
    <pressKey keys ="Hello World" />
    <sleep time="3000"/>
    <captureScreen
filename="\screenshots\Today.bmp"/>
    <clickMouse x="225" y="15"/>
  </test>
</testsuite>
```

Figure 3. XML Test Script

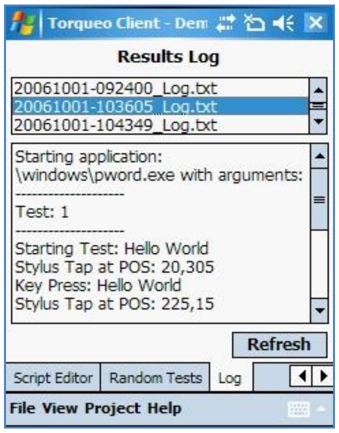
Table 3. Scripted Actions

Action	Description
clickMouse	Simulates a stylus tap in the specified co-ordinates
clickAndHoldMouse	Simulates a right click
sleep	Pause tests for a specified duration
pressKey	Simulates a key press
pressKeyCode	Simulates a key press using a HEX code
captureScreen	Captures the current screen
scanbarcode	Invoke the barcode scanner if present on the device
batchInput	Reads an input file and sequentially inputs each row as separate data
warmboot	Resets the device
padTXTfile	Creates a text file to a specified size
padXMLfile	Creates an XML file to a specified size
writeToDatabase	Writes data to a database
makePhoneCall	Makes a phone call to a specified number
sendSMS	Sends an SMS to a specified number
connectToWeb	Opens the Web browser and navigates to a specified URL

Torqueo does not know or maintain the state of the application under test. It simply performs input as specified in the test XML script. All actions performed by the tool (both randomly generated and scripted input) are logged in a text file as shown in Fig. 4 and Fig. 5. This log file can converted to an XML script and executed again if any errors were found during a random test.

```
Starting application: \windows\pword.exe with
arguments:
--------------------
Test: 1
--------------------
Starting Test: Hello World
Stylus Tap at POS: 20,305
Key Press: Hello World
Stylus Tap at POS: 225,15
Stylus Tap at POS: 225,15
Stylus Tap at POS: 225,15
Stylus Tap at POS: 225,15
Completed Test: Hello World
--------------------
All Tests Completed
--------------------
```

Figure 4. Log file generated at the end of each test



```
20061001-092400_Log.txt
20061001-103605_Log.txt
20061001-104349_Log.txt
```

```
Starting application:
\windows\pword.exe with arguments:
--------------------
Test: 1
--------------------
Starting Test: Hello World
Stylus Tap at POS: 20,305
Key Press: Hello World
Stylus Tap at POS: 225,15
```

Figure 5. Torqueo Log Screen

Torqueo has been successfully used to perform the following types of tests:

1. Reproduce defects that appeared to be random in nature. Once they were reproduced the log could be analysed for what actions were performed before the defect occurred.
2. Soak testing of applications, by performing scripted tests for up to 3 consecutive days.
3. Security Stress Testing an Application in Kiosk mode

V. Advantages and Limitations

Torqueo has a few advantages over previous work. The main advantages are that it runs solely on the device and does not need to be connected to a PC and it runs on any device that has the .NET Compact Framework v2.0 installed. This allows for testing of many applications on many different devices with different form factors. Also, the same XML test can be played on many different devices.

There are also several limitations currently present in *Torqueo* including the inability to maintain state of what application is currently running on the device and what form is currently in focus. Also, while there is functionality to simulate stylus input, there is no functionality to properly record the stylus input as no support for system level input hooks are provided for recording stylus taps in the Windows CE SDK. The way around this problem is to develop a screen driver that supports stylus capture functionality or to capture the mouse events on a clear (opaque) window form and pass these events to the *mouseClick* function within *Torqueo*. Another limitation is that *Torqueo* relies on .NET Compact Framework being installed on the device. However, this will not be a problem for newer devices as the .NET Compact Framework is pre-installed in the ROM on all Windows Mobile 6.1 devices and above.

VI. CONCLUSION

In its current state, *Torqueo* is capable of performing automated stress tests to determine the robustness of Windows Mobile applications running on various device types including PDAs, Mobile Phones, MP3 Players and Embedded PCs. However, work is currently continuing on *Torqueo* not only to overcome the limitations mentioned above but also to enhance the way it performs current tasks. *Torqueo* is currently an intrusive tool that runs on the same device as the application under test. While there are advantages to running the test tool on the device, running multiple tests simultaneously on different devices requires test analysts to perform analysis on multiple individual log/results files. Once *Torqueo* is changed to a lightweight agent running on a device connected to a desktop PC, *Torqueo* will allow tests be executed, test results to be collected and analyzed from a central location. In addition to this agent based architecture, it is planned that .NET Reflection will be used to map GUI objects, and artificial intelligence techniques will be used to generate and execute tests.

Adding this functionality into *Torqueo*, will allow testers and developers to perform automated GUI stress testing throughout development and will ultimately result in stable, robust and high quality mobile applications

REFERENCES

[1] J. Bo, L. Xiang, and G. Xiaopeng, "MobileTest: A Tool Supporting Automatic Black Box Test for Software on Smart Mobile Devices." Second International Workshop on Automation of Software Test, 2007. Minneapolis, MN: IEEE Press, 2009, doi: 10.1109/AST.2007.9

[2] Central Intelligence Agency. Country Comparison :: Telephones - mobile cellular. 2005. https://www.cia.gov/library/publications/the-world-factbook/geos/xx.html (accessed 08 15, 2009).

[3] W. H. Chong, "iDEN Phones Automated Stress Testing." World Academy of Science, Engineering and Technology 23, 2006: pp. 12-16.

[4] W. H. Chong, "iDEN Smartphone Embedded Software Testing." Proceeding of the Fourth International Conference on Information Technology, 2007. ITNG '07. Las Vegas, Nevada, USA: IEEE Press, 2007, pp. 872-873, doi: 10.1109/ITNG.2007.103

[5] M. Fewster and D. Graham, Software Test Automation. Harlow, Great Britain: Pearson Education Limited, 1999.

[6] S. F. Gilani, M. Gillespie, J. Hart, B. K. Mathew, and A. Olsen, .NET Reflection Handbook. Birmingham, United Kingdom: Wrox Press, 2002.

[7] A. M. Memon, "Using Tasks to Automate Regression Testing of GUIs." Proceedings of The IASTED International Conference on ARTIFICIAL INTELLIGENCE AND APPLICATIONS . Innsbruck, Austria: IASTED, 2004.

[8] A. M. Memon, M. E. Pollack, and M. L. Soffa, "A Planning-Based Approach to GUI Testing." Proceedings of The 13th International Software/Internet Quality Week. San Francisco, California, 2000.

[9] A. M. Memon, M. E. Pollack, and M. L. Soffa, "Plan Generation for GUI Testing." Proceedings of The Fifth International Conference on Artificial Intelligence Planning and Scheduling. Breckenridge, Colorado: AAAI Press, 2000, pp. 226-235.

[10] Microsoft. Hopper Test Tool. 08 28, 2008. http://msdn.microsoft.com/en-us/library/bb158517.aspx (accessed 08 15, 2009).

[11] S. She, S. Sivapalan, and I. Warren, "Hermes: A Tool for Testing Mobile Device Applications." Australian Software Engineering Conference 2009. Gold Coast, QLD, Australia: IEEE Press, 2009, pp. 121 – 130, doi: 10.1109/ASWEC.2009.17

[12] M. Sutton, A. Greene, and P. Amini. Fuzzing: Brute Force Vulnerability Discovery. New Jersey, United States: Pearson Education, 2007.

[13] M. Ye, B, Feng, Y. Lin, and L. Zhu, "Neural Networks Based Test Cases Selection Strategy for GUI Testing." Proceedings of the 6th World Congress on Intelligent Control and Automation. Dailian, China: IEEE Press, 2006, pp 5773 – 5776, doi: 10.1109/WCICA.2006.1714182