Challenges in GUI Test Automation

G. Mohan Doss Gandhi and Anitha S. Pillai

***Abstract*—Day by day, software testing becomes very crucial, since the applications are developed in different languages on different OS and environments. Also, the complexity of the software is getting increased. Organizations are adapting agile development to deliver the functionality as quickly as possible. The difficulty in software testing stems from the complexity of software. Regardless of complexity, we need to deliver a high quality on time. Test automation is one of the robust and fastest solutions for achieving quality in complex environment. This paper analyzes the various methods of identifying /recognizing UI controls in GUI Test Automation. It also describes the advantages and disadvantages over Traditional solutions and the solutions implemented on GUIRobo. This paper is a follow up to the "Software Test Automation Using GUIROBO" published on ICCTD 2011 conference.**

***Index Terms*—Test automation, challenges, GUIRobo.**

I. Introduction

Software Testing is the process of interacting with software to evaluate its quality. Testing must be performed in order to ensure that a software program, application or product sufficiently meets all the envisioned business and technical requirements Due to the increased pressure, the testers are forced to release the applications or products more quickly. Undue stress on the testers and manual errors during testing can be avoided by automating the GUI testing process. Automation is the use of strategies, tools and artifacts that augment or reduce the need of manual or human involvement or interaction in unskilled, repetitive or redundant tasks. requirements. The toughest part of automation is interacting with Device under test especially with Graphical User Interface (GUI).

Today‘s software managers and developers are being asked to turn around their products within ever-shrinking schedules and with minimal resources.

Ref. [1] But challenge in delivering quality products is due to the increased pressure to release applications or products more quickly. Even if the products have to be delivered quickly we cannot compromise on the testing methods. Still testers are committed to deliver the best quality products and should also provide assurance to users that the product will perform as promised.

In order to increase the automation, then the Software should be Testable. A testable product ensures complete execution of the test scripts. Also good test coverage is applied; most of the severe defects will be uncovered and fixed before the product is released.

Manuscript received September 5, 2013; revised November 15, 2013.

G. Mohan Doss Gandhi is with Microsoft India (R&D) P.Ltd, Microsoft Campus, Gachibowli, Hyderabad, India (e-mail: mogandhi@ microsoft.com).

Anitha S. Pillai is with Hindustan University, Chennai, Department of Computer Application, India (e-mail: mca@hindustanuniv.ac.in).

II. Manual Testing

Though practice of manual testing is still being used, it has significant weaknesses. The primary weakness is once a problem is found; it is difficult or impossible to reproduce the defect because the tester does not follow a pre-defined sequence of events.

Some of the weaknesses of manual testing are [2]:

1) Manual techniques cannot provide the kind of intense simulation of maximum user interaction over time. Humans cannot keep the rate of interaction up high enough and long enough.
2) 2. Manual testing does not provide the breadth of test coverage of the product features/commands that is needed. People tend to do the same things in the same way over and over so some configuration transitions do not get tested.
3) Manual testing generally does not allow for repeatability of command sequences, so reproducing failures is nearly impossible.
4) Manual testing does not perform automatic recording of discrete values with each command sequence for tracking memory utilization over time – critical for detecting memory leaks.

III. Benefits of Test Automation

Automated testing can provide several benefits when it is implemented appropriately. Automation is a good way to cut down time and cost.

The significant benefits of automated tests are [3]:

- Production of a reliable system.
- Improvement of the quality of the test.
- Reduction of the test effort.
- Testing a large test matrix (different languages on different OS platforms).
- Allows for repeatability of command sequences, so reproducing failures is nearly impossible.
- Automated Tools run tests significantly faster than human users.

The further section discuss abut GUIRobo, the Challenges in Test automation, Traditional solution for overcoming challenges in GUI Test automation and how GUIRobo handles all the challenges in GUI Test automation.

IV. Challenges in Test Automation

Due to the growing complexity of the software, it is impossible to automate GUI applications for 100%. But if we overcome some challenges at least we can automate it for 80%. Multiple challenges in GUI test automations are:

- Application window name is dynamically Changing.
- Controls which don't have proper and unique text property. (E.g. Text and Combo box)
- Mixed up of Managed and Unmanaged UI controls.
- Applications developed in multiple languages and multiple OS.
- Customized controls and Owner draw controls.
- Applications that are using third party controls like source grid control.
- Class names are dynamically changing.
- Control names are dynamically changing.
- Controls which don't have proper Z order.
- More visual controls.
- Support for Win32 Controls.
- Synchronization issue between tool and Device under Test.

V. Approach in Current Automation Tools

A number of Automated Tools have been developed for GUI-based applications. However for many of the companies that purchase these tools, it will not help them to completely automate their testing. Test scripts are either developed or captured using record and play approach.

There indeed are many tools that allow scripts to be recorded and then played back, using screen captures for verification.

Especially if controls have any testability challenges or do not have the unique testability properties, automate the controls based on the coordinate position. Though it is recorded, testers need to modify the scripts to handle the various verification points.

The problem that always crops up is that the layouts are changed, invalidating the screen captures and then the interface controls change making playback fail. Now the scripts must be re-recorded from scratch. Record and playback tools provide an easy way to create throwaway test suites. Test creation should be a cumulative process, with parts of existing tests being recycled to make new tests.

Hence record and play approach is failed in many cases. Though we spent so much money for automation tool, still have to do most of their testing manually. It forces the tester to create their own utility and develop test suites to run their automation. If you spend most of your early testing time writing test scripts and creating test harness, you will delay findings bugs until later, when they are more expensive.

In an ideal world, testers should be able to start test creation at the same time as the development begins, using the requirements for the test design. Most of the commercially available tools are not as good as for tester to start their test immediately. Most of the stress tool is not doing what it is supposed to do [4].

In order to start GUI Stress immediately, we require a routine (test harness or executable) for running test in repeatable fashion.

Hence the tester has to develop his own logic for running stress in repeatable mode. Apart from the test scripts, the resources needed to write and test the code for the tests. Another major challenges for GUI test automation are maintainability and reliability. But many testers do not have strong programming skills to create.

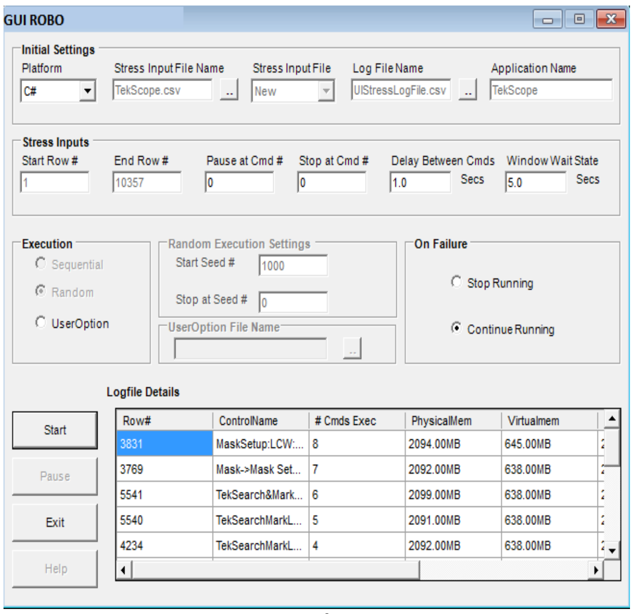
This combined with the repetitive nature of much testing, leads people to use record and playback techniques. GUI Stress tests in particular should be flexible because of the frequency of changes in a developing application's interface.

Typically types of errors uncovered by stress testing include memory leakage, performance problems, lacking problems, concurrency problems, excess consumption of system resources and exhaustion of disk space. In order to find the showstopper defects such as memory leaks, resource leaks land crash defects, the tool should execute the test suites in different modes (random and sequential modes) for long period of time. A good tester will always try to reduce the repro steps to the *minimal steps* to reproduce; this is extremely helpful for the programmer who has to find the bug.

Since every tool has its own limitation and challenges to address the testability issues, it is better to use a combination of multiple technology to take the maximum returns out of test automation investment.

It can be possible only if tool has to handle all the testability challenges efficiently.

VI. What Is GUIROBO?


GUI ROBO

Initial Settings
Platform: C#
Stress Input File Name: TekScope.csv
Stress Input File: New
Log File Name: UIStressLogFile.csv
Application Name: TekScope

Stress Inputs
Start Row #: 1
End Row #: 10357
Pause at Cmd #: 0
Stop at Cmd #: 0
Delay Between Cmds: 1.0 Secs
Window Wait State: 5.0 Secs

Execution: Sequential [✓ Random] UserOption

Random Execution Settings
Start Seed #: 1000
Stop at Seed #: 0

UserOption File Name:

On Failure: Stop Running [✓ Continue Running]

Logfile Details

Row#	ControlName	# Cmds Exec	PhysicalMem	Virtualmem
3831	MaskSetup:LCW:...	8	2094.00MB	645.00MB
3769	Mask->Mask Set...	7	2092.00MB	638.00MB
5541	TekSearch&Mark...	6	2099.00MB	638.00MB
5540	TekSearchMarkL...	5	2091.00MB	638.00MB
4234	TekSearchMarkL...	4	2092.00MB	638.00MB

Fig. 1. Image of GUI ROBO.

GUIRobo [5] is an automated stress testing tool, the test engineer can instruct the tool when to execute a stress test, which tests to run, and how many users to simulate –all without user intervention. It provides an easy to use, consistent and cost- effective way of testing GUI applications developed in C/C++/VC++,C#/VB.Net and WPF.

Initial version of GUIRobo supports GUI Stress testing. For running stress, the tester does not have to write test suites. The tester has to prepare a stress input file for running stress test. Further it will be enhanced to support GUI functional testing, Smoke testing and performance testing.

In order to perform fucntional testing Tester has to write test scripts but they have a flexible option to write their scripts in any .Net plat form.

GUIStress (GUIRobo.exe) takes an input file and will send pseudo-random keystrokes to the (Device Under Test) DUT. The stress input file is an Excel spreadsheet which contains a description of the users interface – what menus, buttons and fields exist. The user has control of how many key strokes (or commands) are sent, which commands are included and a variety of other variables.The tool automatically keeps a log file and on error, it stops. The stress runs can be reproduced from any given point. The image of the GUIRobo is as Fig. 1 given above:

VII. Salient Features of GUIRobo

GUIRobo makes stress testing easy, yet powerful, through its automatic machine resource monitoring feature. Unlike other automated tools, GUIRobo does not require an expensive license and yearly maintenance fees.

The GUIRobo contains the following salient:

1) Tests wide range of environment and languages.
2) Allows to run in different modes such as Sequential Random mode and User Option mode.
3) Automatic resource monitoring monitors Memory usage, Physical memory, Virtual memory USER, GDI for every mouse actions.
4) Helps tester to analyze and reproduce the problem quickly.
5) No need to create test suites and utility to run stress.
6) Simplified way of creating and verifying input files.
7) Ability to run only certain branches and narrow down the defect.
8) Support randomness of field inputs where appropriate.
9) Allows tester to run in command Line mode.
10) Allows tester to provide Delay between the commands and window wait state.
11) Apart from windows standard controls, GUIRobo supports all customized controls such as Source grid 2, Source grid 3 Owner Draw menus, and Owner draw UI Controls.
12) Generates detailed log files.

VIII. Traditional Solutions

Traditionally, we are using the following solutions for identifying the UI control and it has its own advantages and disadvantages as described below:

Solution -1 - Control ID

Advantages:

- Locale –Independent solution.
- Unique over 90 of the time.

Disadvantages:

- Applicable only for MFC.
- In window forms, Control ID is mirror image of HWND.It is different every time we launch the form.

Solution - 2 - Captions and class name

Advantages:

- Best identifier for windows UI.
- Worked well in most cases.

Disadvantages:

- Need to adjust the Caption on all the different locales in which we are testing.
- Not a complete solution as caption is getting changed and does not have caption at all.
- Need to get the Nth instance of combination (Caption+Classname), if captions are not unique.
- Cannot search the UI using class name as trailing portion of the window class name is dynamic. E.g.WindowsForms10.BUTTON.app.3a “3a’may change each time we launch the form .

Solution - 3 – MSAA- AccName and AccRole

Advantages:

- Similar to Captions+Classname,the AccName was localized string.

Disadvantages:

- Acc Role was far from unique.
- Searching MSAA is very slow.
- Tedious to convert to call “windowFromAccessibleObject”to convert HWND.
- Not having enough information to call all windows API.

Solution - 4 – UI automation- automation element and automation ID

Advantages:

- Identify the controls using Automation Elements and Automation IDs.

Disadvantages:

- Searching Automation Element& ID is very slow.
- Solution - 5 – Windows Hierarchy Order.

Advantages:

- Uses child ordering to identify the Windows tree hierarchy.
- Consistent across different versions of OS.

Disadvantages:

- Ordering is not consistent as it keeps changes by adding new control or new level.

Solution - 5 – Support for Win32 controls

The win32 control are identified using User 32 APIs. The handle of the UI controls are identified using class name and caption.

Advantages:

- Identify and automate the controls quickly.

Disadvantages:

- Identify the conrols which have only Unique text or name property.

IX. GUIROBO Solution

GUI Robo uses various automation solution to overcome al l the testability issues. It proposes the following Automation solutions:

- **Application Window Name:** Some application contains the inconsistent window name and it keeps changing to each window it opens.

Hence GUIRobo uses process name to identify the application window name. First it identifies the process name by comparing each process name and then get the appropriate windows title from the process name.

- **Name Property:** In windows forms UI controls, Name property is always unique. Hence

GUIRobo is using Name property for Windows forms UI controls which do not have unique identifier or text property.

- **Control ID:** Unmanged UI controls do not have Name property but it has additional unique property called Control ID. GUIRobo uses Control ID to identify the UI controls which do not have proper Text property. (E.g. Edit box and Combo box).
- **Label Name:** GUIRobo is using the corresponding Label Names for the UI controls which do not have Unique text property. For example unmanaged Combo box and Edit boxes, the text property is keep changing whenever user enters the values on edit boxes or changing the combo item.

Hence GUI Robo first identify the handle of the corresponding Label name and then find the handle of edit or combo box by using get next window API .

- **Controls which do not have proper Z order:** In some unmanaged controls, ’Z’ order won ‘t be proper. If ‘Z’ order is not proer then, idenfying the controls using label name is not possible.

In that case, GUIRobo searches the reference of the privious or next windows. Using that reference window, identifies the required UI controls.

- **Third Party Controls and Owner Draw controls [6]:** GUIRobo is using Hooking process(inject a .NET assembly in another process.) to automate the third party controls, customized controls and owner draw controls. Using handle of the control GUIRobo, access the windows.Forms. Control Property. By Using Control property perform all the click actions, set or get value for the edit or combo items,find the tree nodes of the Tree view control.
- **Mixed UI Controls (Managed/Unmanaged):** GUIRobo distinguishes and handles the mixed UI controls using class name.

For example if class name starts with Window.Forms, then it will be considered as managed controls and if it contains only “BUTTON”, then it will be considered as Unmanged controls .

- **Visual Controls :** GUIRobo uses bitmap comparison for verifying visuals and image controls.It compares pixel by pixel and create a log file if there is any mismatch between the bitmaps.
- **Dynamic Class name Identification:** In .Net, class names are dynamically changing. Especially, the prefix (WindowsForms10) and suffix (app.3a) of the class name keeps changing.

Ex: WindowsForms10.BUTTON.app.3a.

Hence GUIRobo identifies the prefix and suffix of the class name and forms the class name for each controls.

- **Synchronization:** GUIRobo uses wait sate/delay for each window to resolve the synchronization issue. Allows user to configure the Window’s wait state according to the time required to open the window.

GUIRobo also supports the delay between each command execution to resolve the synchronization issues between DUT and Tool.

- **Support for Win32 Controls:** GUIRobo uses mixed approach to identify and automate the win 32 UI controls. If controls do not have the unique property, then GUI Robo Identify the bounding rectangle of the controls using UI automation framework. Then execute the click event using User 32 APIs. Similarly GUIRobo uses Keyboard-Event to set the value to edit boxes .
- **Support for WPF Controls [7]:** Since there are no handles concept in WPF controls, GUI Robo uses UI automation framework for identifying the WPF controls. UI controls are identified using either Automation Element Name or Automation ID property. Using UIAutomation framework,

GUI Robo identify theAutomation Element of the specific WPF controls. It searches the elements from Root element. Once Element is identified, GUIRobo uses either Mosue Click event or appropariate supported pattern.

For example, GUI Robo uses *Invoke pattern* for clicking particular WPF button. Similarly uses *ValuePattern* for entering text value.

The advantage is, this technique can be extented to automate Web applications as well .

X. Conclusion

Using GUIRobo we can overcome all the testablity challenges in GUI Test automation and automate more GUI controls. It makes the test process more stable, more efficient and, ultimately, reduces the cost while increasing the quality of the delivered product. It makes your testing more consistent and there is no doubt that GUIRobo adds a great value to overall quality.

References

[1] Achieve Quality through GUI Test Automation in Complex Environments Mohan Doss Gandhi. [Online]. Available: http://www.pnsqc.org/past-conferences/2010-conference/poster-paper-presentations/#10

[2] P. Hegde, P. Hunter, F. Liang, and D. Reynolds. Monkey testing revisited using automated stress testing. [Online]. Available: http://www.agileconnection.com/sites/default/files/article/file/2013/XUS2314150file1_0.pdf

[3] Advantages of Automated Testing. [Online]. Available: http://www.scribd.com/doc/7160453/Advantages-of-Automated

[4] E. Dustin, J. Rashka, and J. Paul, *Automated Software Testing: Introduction, Management, And Performance*, Addison-Wesley Professional, 1999.

[5] G. M. D. Gandhi and A. S. Pillai, “Software test automation using GUIROBO,” in *Proc. International Conference on Computer Technology and Development*, 2011, vol. 1, 3, pp. 641-646.

[6] R. K. Vavilala, *A simple Windows Forms Properties Spy*.

[7] C. Tatar. WPF UI Automation. [Online]. Available: http://www.codeproject.com/Articles/33049/WPF-UI-Automation

Mohan Das Gandhi holds a master’s degree in Applied Electronics from PSG CAS, Coimbatore, Tamilnadu, India and is currently pursuing a doctoral program (Ph.D.) in Software Quality engineering at Hindustan University, Chennai.

He is presently employed in Microsoft India (R&D) P.Ltd, Hyderabad, as a senior test lead. Overall, he has 15+ years of experience in the IT industry. He has vast experience in GUI test automation tool development and has done a lot of research in GUI test automation.

So far Mr. Gandhi has submitted one Technical Paper in ICCTD2011 and it has been published on ASME press and three poster papers to PNSQC.