I.J. Intelligent Systems and Applications, **2015, 12, 68-74**
Published Online November 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2015.12.07

Computer Vision based Automation System for Detecting Objects

Rajibul Anam
Samsung R&D Institute Bangladesh, Dhaka, 1205, Bangladesh
Email: rajibul.an@samsung.com

Mostafizur Rahman, Mohammad Obaidul Haque and Md. Syeful Islam
Samsung R&D Institute Bangladesh, Dhaka, 1205, Bangladesh
Email: topucse06@gmail.com, {obaidul.h, syeful.islam}@samsung.com

***Abstract*—**Software Quality Assurance Testing time computer vision based automation tools are used to test the window based application and window based application is combined of many objects. Among them most of the automation tool detect window objects by comparing images. Most of the objects are visible in the window screen but some objects which are not visible to the screen at the first time. Proper interaction with the window application hidden objects get visible to the screen like dropdown list item, editor text object, list box item and slider. With the automation tools these hidden objects cannot be searched directly. In this paper proposes some methods which will enhance the automation tools to access the window application hidden objects.

***Index Terms*—**Automation, test case, blackbox testing, vision based, window application.

I. Introduction

Software Quality Assurance (QA) is one of the critical areas of software development process. After the co-work with developer and designers, QA ensures the correctness of the operation by testing the software through different type of test cases [1]. Many methods have been used to test the software and among these methods Black Box and White Box Testing are very common. The Black Box testing consists of Specification and Experience based testing which checks the entire software operation [1-2]. The White Box testing follows the Structure based testing which checks the software process flow [1-2]. Testing time most of the actions or steps are followed by mouse and keyboard events which change the program flows and interfaces [2]. Manual testing operate by human which executes series of steps and check for the specific output which has chances of error [1],[3]. The Automation tool executes series of steps according to the code instruction, which executes test steps faster than human and less error [3]. Moreover, automation has been used for Black Box testing because it follows specific test steps and expects for target results.

Graphical User Interface (GUI) testing purpose many type of automation tools have been used such as Pesto [4], DEVSimPy [5], Watir [1], Selenium [7], Sikuli [6],[8] which uses the vision based screen image detection and shortcut keys to track the screen objects. Reusability and smooth execution are essential for the automation tool [6]. QA testing steps/actions executes easily with these automation tools by tracing image, mouse and keyboard events. Window application hidden or not visible objects like specific text object in the editor, dropdown list object, multi tab scroll object and also some complex steps which cannot executes by image based tools like select object from the list box and slider. Considering these difficulties here, focuses on how to access hidden objects, change the object display values accurately and enhance reusability.

This paper proposes some methods which will enhance image based automation tools to discover hidden objects from the window based applications. The propose methods uses shortcut key and image object to select screen object from the window visible screen. Through proper interaction with the visible objects, the hidden object appears on the screen.

This paper is arranged as follows. Section 2 provides brief review of other automation tools for GUI testing. Section 3 describes about the proposed methods. Section 4 details discussion and results of the proposed methods and finally conclusion is on section 5.

II. Related Works

Software QA testing there are many type of automation tools are available. Automation tools track the objects by screen object position, screen object image and screen object source name. Actions/steps are executes by cursor movements, clicks, drag-and-drop and keyboard input events. Tools like Sikuli, Robot and Pesto uses image and shortcut key to access the object.

A. Sikuli Framework

Sikuli is an open source GUI vision based automation (visual testing) tool, which searches the screen object using screenshot [8-10]. The IDE permits users to take a screenshot of the object (GUI elements) such as button, icon, dialog box and the IDE run time detects the object to direct the mouse and keyboard events [11-13]. Figure 1 illustrates the Sikuli Framework, where built-in modules are available like find, click and keyboard events [14]. There are more modules available which cannot be access from the IDE directly. It has the Application Programming Interface (API) for testing and developing the library. It is a platform independent framework.

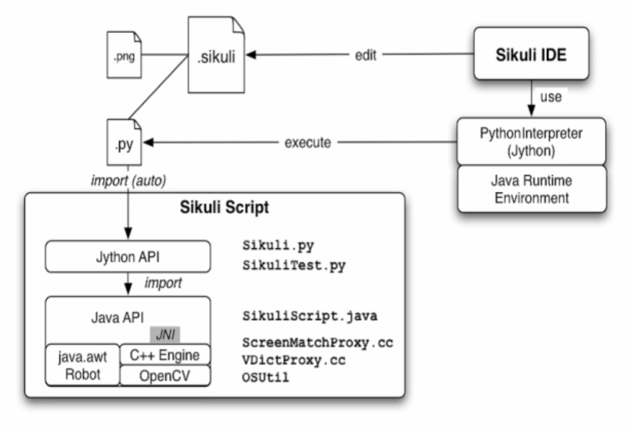


Fig.1. Sikuli Framework

B. *Robot Framework*

Robot Framework is a generic testing automation system to test Acceptance Test-Driven Development (ATDD) [15]. ATDD is a process where developers and testers discuss the demands required by the customers to come with the acceptance test before development. The acceptance test provides the functional importance of the software [15].

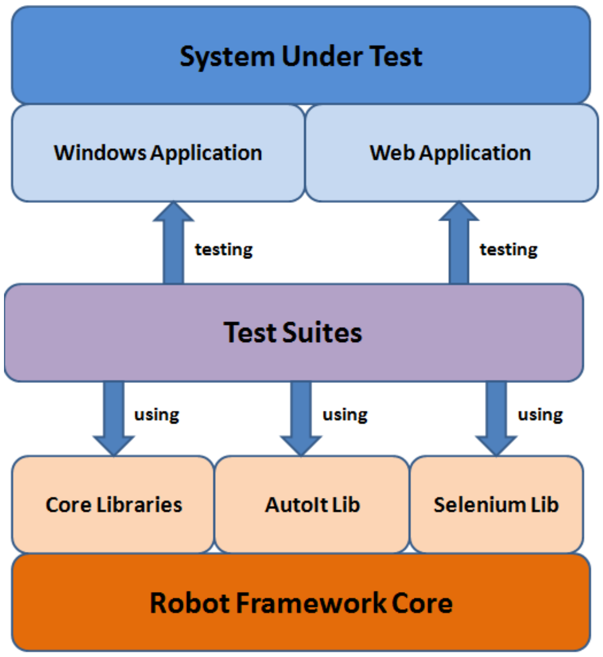


Fig.2. Robot Framework

The aim of the acceptance tests is to justify the requirements by providing examples for each test. The examples can be tested to prove compliance. The script language is written using plain English natural commands called Keywords [9]. Keywords are common like methods in programming language. Natural command keywords make the tests more readable and easy to understand even for non-coders. This framework script writing is extended to Python (can run also on both Ironpython and Jython) or java. The developers can use the existing syntax to create the script or can create own syntax. Robot framework uses for GUI testing and system resource management, but only java based software can be tested. It generates auto report of the testing as html and text format. It has the API for testing and developing the library.

III. PROPOSED METHODS

Many automaton systems are available and most of them are uses vision based algorithm to search the GUI objects. These systems take the screenshot of the window first; then select the target object and interact with the application by mouse or keyboard events. These tools used to search the GUI window objects like toolbar button, menu item, icon and dialog box [14]. The searching arbitrary depends on the screenshot and the target object image. The target object image do not matches with the screenshot image, then the system could not search the target object in the screen. In this case either user may want to search an object which is not visible (hidden object) to the screen or the object is not available in the testing software. So the automation system will not be able to search the object until the target object gets visible on the screen.

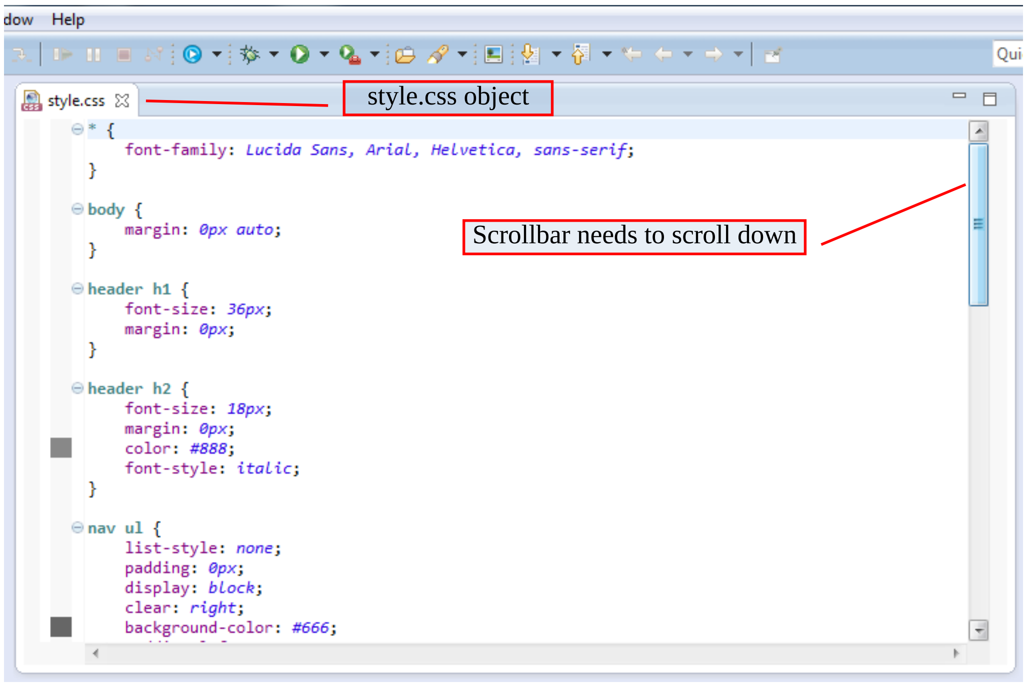
Current approaches required entering an image as query to search the object. If searching for a hidden object, then the automation system cannot trace the object which is a limitation of the system.

The proposed method searches hidden objects like item in the editor, dropdown list object, multi tab scroll object and complex object like slider positions. In addition shortcut key used instead of image object to take mouse focus on the object. The Editor Scrollbar Object Selection method uses to search the hidden object from a scrollbar affiliated object. The Dropdown List Object Selection method is applicable to search the hidden object from the dropdown list. The Multi Tab List Object Selection method is valid to search the hidden object from the multi tab list box and Slider Positon Selection method is applicable to search the slider, puts the slider values according to the user selection. Below sections discuss details of the proposed methods.

A. *Editor Scrollbar Object Selection*

The automation tool basically executes command according to the image and shortcut key based actions. But hidden objects are not visible to the screen and could be visible if the system searches for the hidden objects [14]. Figure 3 shows the editor screen with scrollbar object, where a text editor is opened. The automation tool needs to search the figure 4 target object (search in the screen) in the (figure 3) text editor. It is a very complex scenario to scroll down the scrollbar using the mouse through automation [13]. There is no measurement scale for calculating the scrollbar to scroll down. Need to find the target object (figure 4) which is not visible on the screen, by searching the hidden objects, the target object can be found. To solve this problem, proposed the Editor Scrollbar Object Selection method.

Figure 5 line 1-2 searches the *mainobject* (the style.css object) and put the cursor focus on the object. Line 3 put the cursor at the beginning of the editor. Line 4-9 searches for the *targetobject* (figure 4), if do find, then go to the next line until it reaches to the *maximumline* (maximum number of lines in the page), if finds the *targetobject* (figure 4), then select (click) the object. The scrollbar cannot be used directly (can access it but cannot scroll it as needed) and with this method can search the hidden object from the screen. Figure 6 shows where the hidden object gets visible and *targetobject* is found by using this method.



```
* {
    font-family: Lucida Sans, Arial, Helvetica, sans-serif;
}

body {
    margin: 0px auto;
}

header h1 {
    font-size: 36px;
    margin: 0px;
}

header h2 {
    font-size: 18px;
    margin: 0px;
    color: #888;
    font-style: italic;
}

nav ul {
    list-style: none;
    padding: 0px;
    display: block;
    clear: right;
    background-color: #666;
```

Fig.3. Screenshots of a Scrolling Object

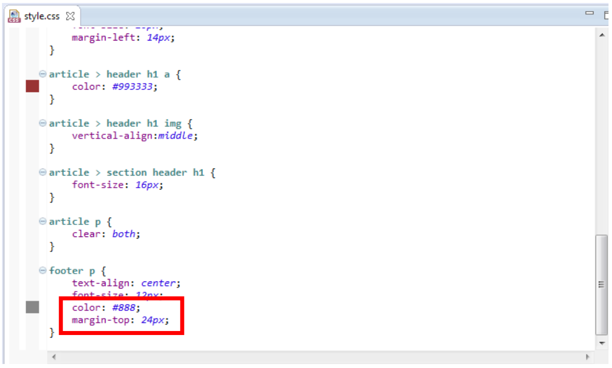
```
color: #888;
margin-top: 24px;
```

Fig. 4. Hidden Target Object

```
Input:
        mainobject is an image or text object to focus on the object;
        maximumline is the maximum number of lines in the scroll; object;
        targetobject is an image object;
Output:
        targetobject get selected;
Variables:
        screenimage is the desktop screen capture image;
        onelinedown is an keyboard value to move down the cursor next line;

ScrollbarObjectSelection(mainobject, maximumline, targetobject)
1.      If mainobject matched with screenimage Then
2.        Click on the mainobject;
3.        Put cursor to the beginning of the editor;
4.        While until found the targerobject
5.          Move the cursor onelinedown;
6.          If targetobject matched with the screenimage Then
7.            Click the targetobject in the screen;
8.          If maximumline is over Then
9.            Cannot found the targetobject;
10.     Else Cannot found the targetobject;
11.     End
```

Fig.5. Editor Scrollbar Object Selection Method



```
    margin-left: 14px;
}

article > header h1 a {
    color: #993333;
}

article > header h1 img {
    vertical-align:middle;
}

article > section header h1 {
    font-size: 16px;
}

article p {
    clear: both;
}

footer p {
    text-align: center;
    font-size: 12px;
    color: #888;
    margin-top: 24px;
}
```

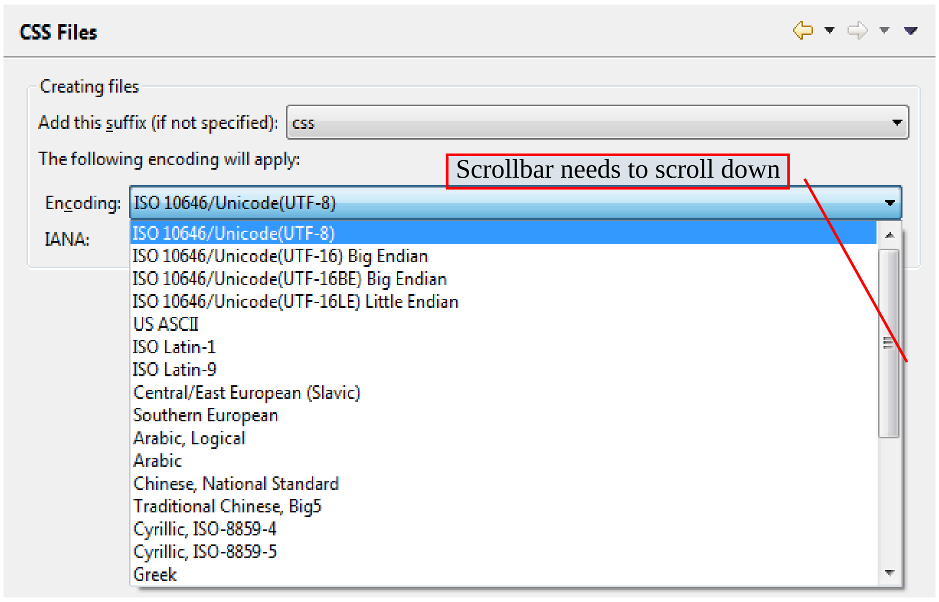
Fig.6. After Scrolling the Screen

B. Dropdown List Object Selection

Dropdown list objects are used in the window based application which needs to be tested. But the dropdown list items cannot be access properly to discover the hidden objects [14].

Figure 7 shows the dropdown list. The automation tool can access the dropdown list but if the list is long, the list object do not appears in the screen, needs to scroll down the scrollbar, that time system cannot take control of the scroll bar to pull it down. So to overcome this issue introduces the Dropdown List Object Selection Method.

Figure 8 shows the target object which needs to search from the dropdown list. Figure 9 shows the method; line 1 where *shortcutkey* is used to select the dropdown list object. Line 4-9 searches for the *targetobject* from the list, goes to next object in the list, if not found *targetobject* then again search until it reaches to the *maximumlist*. Figure 10 shows the *targetobject* matched with screenimage. This method searches the entire hidden object from the list and checks for the target object.


CSS Files

Creating files

Add this suffix (if not specified): css

The following encoding will apply:

Encoding: ISO 10646/Unicode(UTF-8)

IANA:

ISO 10646/Unicode(UTF-8)
ISO 10646/Unicode(UTF-16) Big Endian
ISO 10646/Unicode(UTF-16BE) Big Endian
ISO 10646/Unicode(UTF-16LE) Little Endian
US ASCII
ISO Latin-1
ISO Latin-9
Central/East European (Slavic)
Southern European
Arabic, Logical
Arabic
Chinese, National Standard
Traditional Chinese, Big5
Cyrillic, ISO-8859-4
Cyrillic, ISO-8859-5
Greek

Fig.7. Screenshot of Dropdown List

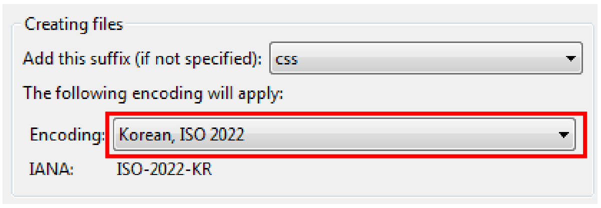
Korean, ISO 2022

Fig.8. Target Object

```
Input:
        shortcutkey is keyboard value to take focus of the object;
        maximumlist is the maximum number of lists in the
        dropdown list;
        targetobject is an image object;
Output:
        targetobject get selected;
Variables:
        screenimage is the desktop screen capture image;
        onelinedown is an keyboard value to move down the cursor
        next line;

DropdownListObjectSelection(shortcutkey, maximumlist,
targetobject)
1.      If shortcutkey works to select the object Then
2.        While until the maximumlist
3.          Move the cursor onelinedown;
4.          If targetobject matched with screenimage Then
5.            Click the targetobject in the screen;
6.          ElseIf maximumline is over Then
7.            Cannot found the targetobject;
8.      Else Cannot found the targetobject;
9.      End
```

Fig.9. Dropdown List Object Selection Method


Creating files

Add this suffix (if not specified): css

The following encoding will apply:

Encoding: Korean, ISO 2022

IANA: ISO-2022-KR

Fig.10. Screenshot of the Dropdown List with Target Object

C. *Multi Tab List Object Selection*

QA there are some tasks which need to interact with one window but multi tab objects [13]. Figure 11 shows an example of the multi tab objects. At first figure 11 (1) (sample) will be selected, then figure 11 (2) (Web App) will be selected from the list, after that figure 11 (3) (item list) will be selected, then figure 11 (4) scrolls down the scroll bar and selects the target object (figure 12).

The proposed Multi Tab List Object Selection method solves the multi tab, hidden and scrolls bar objects detection problem. Figure 13 shows the Multi Tab List Object Selection method where line 1-2 selects the *mainobject* (figure 11, object 1). Line 3-9 selects the tab object and searches (figure 11, object 2) for the *firstkeyinfo* image object until it reached to the *maximumfirstlist*. Line 10-17 select next tab object (figure 11, object 3), search for the *tergetobject* (figure 12), if *targetobject* not found then goes to the next list item until it reached to the *maximumsecondlist*. Figure 14 shows the target object found using this method.

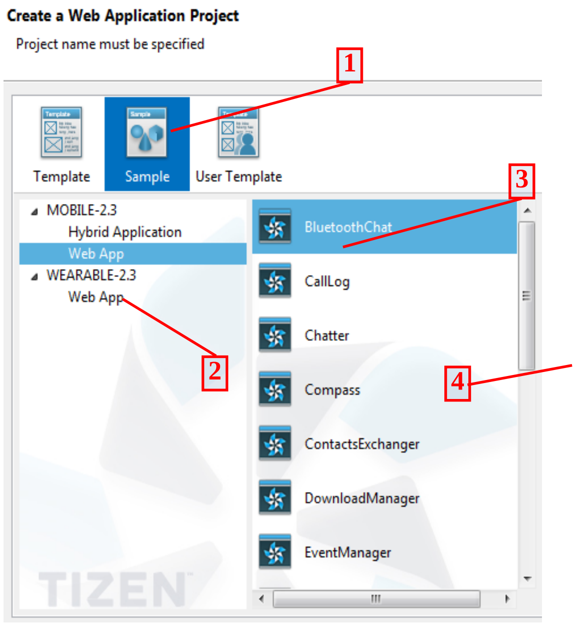


Fig.11. Multi Tab List Object Screenshot

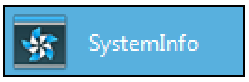


Fig.12. Hidden Target Object

```
Input:
        mainobject is an image object;
        firsttab is a keyboard tab value;
        firstkeyinfo is an image object;
        maximumfirstlist is the maximum number of item in the
object;
        secondtab is a keyboard tab value;
        secondkeyinfo is keyboard cursor down value;
        maxiumsecondlist is the maximum  number of item in the
object;
        targetobject is an image object;
Output:
        targetobject get selected;
Variables:
        screenimage is the desktop screen capture image;
        onelinedown is an action variable to move down the cursor
        next line;

Multitabobjectselection(mainobject, firsttab, firstkeyinfo,
maximumlistfirst, secondtab, secondkeyinfo, maxiumlistsecond,
targetobject)
1.      If mainobject matched with screenimage Then
2.        Click on the mainobject;
3.        If firsttab is true Then
4.          Press tabkey;
5.          While until maximumfirstlist
6.            If firstkeyinfo matched with screenimage Then
7.              Click the firstkeyinfo in the screen;
8.            Elsif maximumfirstlist is over Then
9.              Cannot found the firstkeyinfo;
10.       If secondtab is true Then
11.         Press tabkey;
12.         While until maxiumsecondlist
13.           Press secondkeyinfo;
14.           If targetobject matched with screenimage Then
15.             Click the targetobject in the screen;
16.           If maxiumsecondlist is over Then
17.             Cannot found the targetobject;
18.     Else Cannot found the targetobject;
19.     End
```

Fig.13. Multi Tab List Object Selection Method

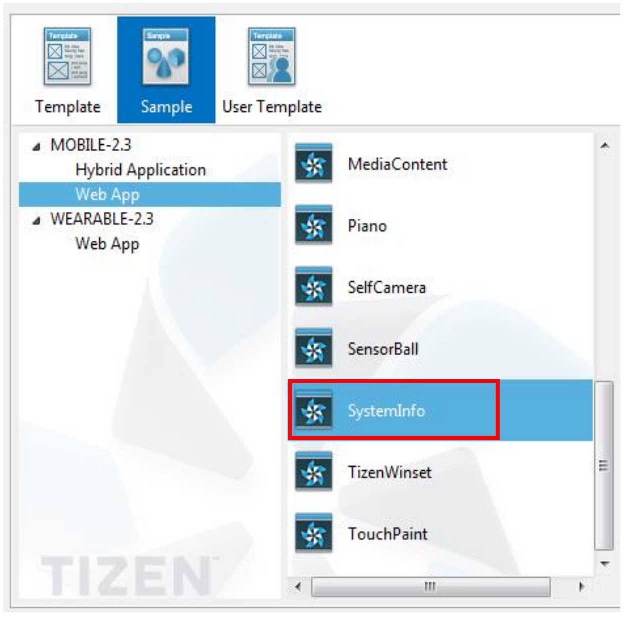


Fig.14. Screenshot of Multi Tab List Object with Target Object

D. *Slider Position Selection*

Slider is a window application object which has no onscreen values (from where to drag and drop) like scroll bar. But for the QA testing purpose automation tool needs to access and change the value of the slider [13]. Figure 15 shows the slider where it needs to set as Debug (figure 16). While testing time system does not have any knowledge which position the slider would be.

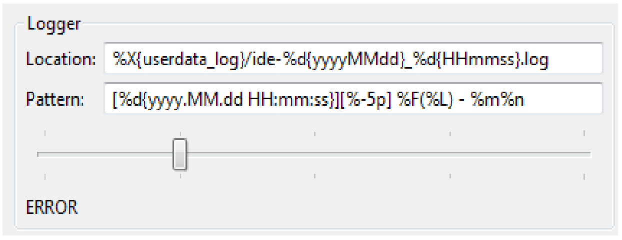

Logger
Location: %X{userdata_log}/ide-%d{yyyyMMdd}_%d{HHmmss}.log
Pattern: [%d{yyyy.MM.dd HH:mm:ss}][%-5p] %F(%L) - %m%n
ERROR

Fig.15. Screenshots of Slider Object

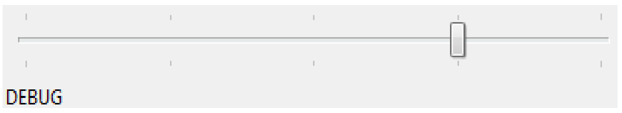


Fig.16. Target Object

The proposed Slider Position Selection method can overcome this problem and put the slider position according to the automation tools instruction. Figure 17 illustrates the algorithm line 1-2 take focus on the slider object; line 3 makes the slider lowest sliding position. Line 3-6 increases the slider positions according to the *keyinfo* value until it reached to the *keymove* and checks the *targetobject* with the *screenimage*, if the *targetobject* matched then stop.

```
Input:
        mainobject is an image object;
        keymove is the maximum slider value;
        keyinfo is a keyboard key move value;
        targetobject is an image object;
Output:
        targetobject get selected;
Variables:
        screenimage is the desktop screen capture image;
        onelinedown is an action variable to move down the cursor
        next line;

scrollslider(mainobject, keymove, keyinfo, targetobject)
1.      If mainobject matched with screenimage Then
2.        Click on the mainobject;
3.        Scroll slider to lowest value;
4.        While until keymove
5.          Press keyinfo;
6.          If targetobject matched with screenimage Then
7.            targetobject object found;
8.      Else Cannot found the targetobject;
9.      End
```

Fig.17. Slider Positon Selection Method

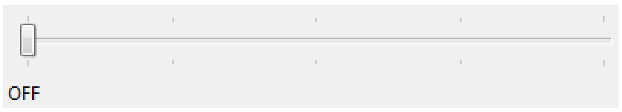


Fig.18. Slider Position with Minimum Value

IV. RESULTS AND DISCUSSION

GUI Automation system executes action according to the instructions (code). Basically two types of events work in the GUI automation. One is keyboard event and another is mouse event. Automation system runs the code to execute commands which interacts with GUI testing system. To generate mouse or keyboard events these systems record the user actions, automatic generates code for automation. And image based frameworks do not have this facility, developer needs to write code. Table 1 shows the comparison criteria of automation frameworks. The sikuli [9] is an image based automation framework. The TestComplete [16], TestPlant [17] and Squish [18] are test recoding based framework, all the events occurs according to the screen window object position (resolution size). If the object posion gets change or mismatched then select different objects and generate error.

Table 1. Comparison of Automation Framework Criteria

	Sikuli	**TestComplete**	**TestPlant**	**Squish**
Open Source	Yes	No	No	No
App code required	Yes	Yes	Yes	Yes
Platform Independent	Yes	No	Yes	Yes
Slider Interaction	No	Yes	Yes	Yes
Hidden Object Identification	No	No	Yes	Yes
Image Based	Yes	No	No	No
Screen Position Dependent	No	Yes	Yes	Yes
Test Recording	No	Yes	Yes	Yes

Test Recording based framework can access the hidden objects, if the screen window objects prosition remain fixed on second run time. But testing time it is very hard to confirm the window objects position. Developer record the test, execution time window object appears at different position, then developer needs record the steps again which is redundent. Table 2 illustrates the frameworks, if the screen window object position gets change, then cannot interact with the objects to search the hidden objects. Moreover slider object also cannot be traced. But same time the proposed methods are able to interact with the changed screen position window objects. The proposed metods use shortcut keys and image objects to interact with the window objects to search the target hidden objects.

Table 2. Comparison of Framework with Screen Object Position

Criteria		**Sikuli**	**TestComplete**	**TestPlant**	**Squish**	**Propose Methods**
Changed Screen Position	**Hidden Object Identification**	No	No	No	No	Yes
	Slider Interaction	No	No	No	No	Yes

The time complexity depends on flow of the algorithm [19]. If the algorithm uses nested operation then the complexity gets higher. Below table 3 shows the comparison of the Proposed Algorithm (PA) and existing Vision Based Algorithm (VA), where O denotes as growth of a function and n is number of steps. It is clear that VA and PA time complexity are almost same. There is no significant difference between PA (ESOS, DLOS, MTLOS and SPS) and VA. But there are differences on the execution time because of the dependency (wait for the object, interaction methods).

Table 3. Complexity of the Algorithms

Time Complexity of VA	**Time Complexity PA**
$O(n)$ [14]	$O(n)$ ESOS
$O(n)$ [14]	$O(n)$ DLOS
$O(n)$ [14]	$O(n)$ MTLOS
$O(n)$ [14]	$O(n)$ SPS

V. CONCLUSION

GUI automation tools enhance Test Case execution and reduce human efforts. Most of the Black Box Test Cases can be executed with these tools, limitation of the technology, image based automation tools cannot find hidden objects and dynamic appearance of the objects. As a result many Test Cases cannot be executed using GUI animation tools. The proposed techniques have the unique features to identify hidden objects even the window objects screen position gets change. The proposed methods are implemented in real time automation application and can discover the hidden objects smoothly. Currently there are two limitations with these methods. Firstly, takes time to check the hidden list box objects one by one and secondly, needs to specify the list box item number to search the target object. Future plan is to overcome these two limitations and works for complete introducing full testing framework for hidden object detection.

REFERENCES

[1] S. Inderjeet, and T. Bindia, "Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli and Watir", International Journal of Information & Computation Technology, vol. 4, pp. 1507-1518, 2014.

[2] K. Dea-Kwang, and L. Lee-Sub, "Reverse Engineering from Exploratory Testing to Specification-based Testing", International Journal of Software Engineering and Its Applications, vol. 8(11), pp. 197-208, 2014.

[3] L. Maurizio, S. Andrea, R. Filippo, and T. Paolo, "Automated Generation of Visual Web Tests from DOM-based Web Tests", ACM/SIGAPP Symposium on Applied Computing, April, 2015.

[4] M. Leotta, A. Stocco, F. Ricca, P. Tonella, "PESTO: A Tool for Migrating DOM-Based to Visual Web Tests", ACM/SIGAPPSymposium on Applied Computing, April, 2015.

[5] L. Capocchi, J.F. Santucci, T. Ville, "Software Test Automation using DEVSimPy Environment", International Conference on Principles of Advanced Discrete Simulation, May, 2013.

[6] Borjesson, and F. Robert, "Automated System Testing using Visual GUI Testing Tools: A Comparative Study in Industry", "Borjesson2012visual", 2012.

[7] J. Hyunjun, L. Sukhoon, B. Doo-Kwon, "An Image Comparing-based GUI Software Testing Automation System", World Congress in Computer Science, Computer Engineering, and Applied Computing, 2012.

[8] K. Pragya, "Ameliorating the image matching algorithm of Sikuli using Artificial Neural Networks", International Journal of Computer Science & Communication, vol. 5, pp. 1-4, 2014.

[9] http://www.sikulix.com, 12-Jan-2014.

[10] C. Tsung-Hsiang, "Using graphical representation of user interfaces as visual references", The 24th annual ACM symposium adjunct on User interface software and technology, pp. 27-30, 2011.

[11] S. L. M. Jeffrey, "User interface computation as a contextualized approach for introductory computing instruction", The 9th Annual International ACM Conference on International Computing Education Research, pp. 179-180, 2013.

[12] V. Andriychenko, L. Ying-dar, C. T. National, "Automatic Functionality and Stability Testing Through GUI of Handheld Devices", CiteSeerx, 2011.

[13] C. Tsung-Hsiang, Y. Tom, C. M. Robert, "GUI testing using computer vision", CHI 10th Conference on Human Factors in Computing Systems, pp. 1535-1544, 2010.

[14] Y. Tom, C. Tsung-Hsiang, C. M. Robert, "Sikuli: using GUI screenshots for search and automation", The 22nd annual ACM symposium on User interface software and technology, pp. 183-192, 2009.

[15] Http://robotframework, 10-Nov-2014.

[16] Http://smartbear.com/product/testcomplete, 12-Nov-2014.

[17] Http://www.testplant.com, 14-Nov-2014.

[18] Http://www.froglogic.com, 16-Nov-2014.

Authors' Profiles

Rajibul Anam obtained International Diploma in Programming & IT from BRAC Information Technology Institute, Bangladesh, he completed his Bachelor of Information Technology (Honours) (Software Engineering) and Master of Science (Information Technology) by Research, "Mobile Content Adaptation Using Tree-Based Algorithms" from Multimedia University, Cyberjaya, Malaysia. At present he is working as a Senior Software Engineer at the Samsung R&D Institute Bangladesh LTD. He is active in the research field of soft computing. Research Interests in Soft Computing, Artificial Intelligence and Mobile HCI. Email: rajibulanam@gmail.com

Mostafizur Rahman obtained B.Sc. from Bangladesh University of Engineering and Technology (BUET) in 2012. Since then he has been working in Samsung Research and Development Institute Bangladesh (SRBD) as Sr. Software Engineer.

Mohammad Obaidul Haque received MSc from University of Trento in 2009. He worked as research intern in SOA Group, Fondazione Bruno Kessler, Italy from 2010 to 2011 on mobile service computing. Afterwards he joined in Samsung Bangladesh R&D Institute in 2012 and now he is leading to develop apps for mobile media content sharing in UPnP based home network environment and enforcing policy aware service composition on Android and iOS platform. He has successfully managed and delivered 4+ R&D and commercial projects in Samsung and University of Trento. His research interest is Data Mining, Data Analytics, Software Engineering, Cloud Computing, and Database & Knowledge-Base System.

Md. Syeful Islam obtained his B.Sc. and M.Sc. in Computer Science and Engineering from Jahangirnagar University, Dhaka, Bangladesh in 2010 and 2011 respectively. He is now working as a Senior Software Engineer at Samsung R&D Institute Bangladesh. Previously he worked as a software consultant in the Micro-Finance solutions Department of Southtech Ltd. in Dhaka, Bangladesh. His research interests are in Natural Language processing, AI, embedded computer systems and sensor networks, distributed Computing and big data analysis.

Manuscript received January 16, 2009; revised June 21, 2009; accepted July 12, 2009.