International Journal of Computer and Electrical Engineering, Vol. 2, No. 3, June, 2010
1793-8163

Agent based User Interface Test Automation for Silverlight Applications

Dhavachelvan.P, Suresh Joseph.K and Appasami.G

Abstract - **User Interface Test Automation for Silverlight Applications (UITASA) plays a vital role in software industry; especially User Interface Test Automation (UITA) in new technology like Silverlight is a challenging task because of its high security and low accessibility. Agent Technology is intermediate software that provides a better bridge between User Interface Test Automation and Silverlight applications. There are different software agents for each group of controls to do User Interface Test Automation. For example button control agent will take care of all buttons in User Interface Test Automation.**

Index Terms - **User Interface, Test Automation, Silverlight Applications and Agent based User Interface Test Automation.**

I. Introduction

Silverlight applications are used to create platform independent, browser independent interactive and attractive web applications with Dot net supporting languages. Silverlight is Microsoft's new cross browser or delivering richer interactive applications to users over the web. Silverlight 2.0 is Microsoft's second release of the Silverlight platform [18][19]. Silverlight 2's biggest change from Silverlight 1.0 is the inclusion of a compact version of the .NET Framework, complete with the .NET Framework 3.0 Common Language Runtime. By adding .NET to Silverlight, Microsoft makes it easy for .NET developers to reuse their existing programming skills, collaborate with designers, and quickly create rich applications for the Web. One of the key benefits of Silverlight 2 is that it can execute any .NET language, including C# and VB.NET [17][18]. Silverlight 2 ships with a "lightweight" version of the full .NET Framework, which features, among other classes, extensible controls, XML Web Services, networking components, and LINQ APIs. This class library is a subset of the .NET Framework's Base Class Library, which enables the Silverlight plug-in to be a fast and small download. In addition to the .NET Framework classes, Silverlight 2 also ships with a subset of the WPF UI programming model, including support for shapes, documents, media, and WPF animation objects [15][18][19].

Agents will act as domain experts in UI Test Automation. Each agent will observe the environment and react on the environment by perception. Each Agent plays a vital role in UI Test Automation. For example text box agent will take care of all text boxes. It will act as an inter mediator for UI Test Automation and Silverlight applications.

Agents will decide themselves to satisfy their design objectives. Agents are best in splitting record values and send particular values to particular agents. Each agent will take care of group of similar controls. So we can say agents are Inter mediator in UI Test Automation of Silverlight applications.

Dhavachelvan.P, Suresh Joseph.K and Appasami.G are from Department of Computer Science, Pondicherry University, Kalapet, Pondicherry, India. (e-mail: dhavachelvan@gmail.com, sureshjosephk@yahoo.co.in, appas_9g@yahoo.com)

II. Silverlight

Silverlight is Microsoft's new cross browser or delivering richer interactive applications to users over the web. Silverlight 2.0 is Microsoft's second release of the Silverlight platform. Silverlight is a Microsoft .NET new UI Language for attractive web applications. Silverlight is developed my Microsoft on Dot Net 3.0 Frame work. So Silverlight provides security up to the level of Dot Net 3.0 Frame work. But Silverlight controls are written by a language called XAML. Silverlight 2 ships with a "lightweight" version of the full .NET Framework, which features, among other classes, extensible controls, XML Web Services, networking components, and LINQ APIs.

Advantages of Silverlight are:

- Micro Soft new technology for web
- Extension of ASP.NET
- Improved visualization
- XAML language for UI description
- Rich Interactive & Attractive Web Applications
- work in all platforms and in all browsers
- Animated Movies can be sent to client with compressed data
- Rich Multimedia, Audio, Video & Animation Support
- Developing device Independent components
- Combined work of Flash and .Net
- write-once-run-everywhere
- WPF-subset and .NET-subset

III. User Interface Test Automation

Testing of UI is always challenging and has been mostly manual till now. In the next generation of MES applications the testing of GUI needs to be automated as it will save lot of man-hours that is lost in manual testing also it will catch the defects early on in the cycle.

UI Test Automation is a combined process of UI Testing and UI Automation. The data flow of UI Test Automation is

shown in figure 1. Usability Testing is a mode of testing a particular product for its compatibility in terms of use. GUI testing is a commonly known form of Usability Testing of software or website. It is also known as User Interface Testing. GUI testing is a performance related assessment of a software or website in terms of ease of use, versatility, friendliness with focus on the target audience, visual impact and the approach and time taken to progress into the specific purpose.

UI Test Automation is a process of doing the testing of the application using appropriate tools and following various testing methodology. Different tools are available from various vendors with concentration on different type of testing. But the fact also remains that the blending of manual and automated testing methods is the best way to test any application. Testing Automation should concentrate on the following factors: Test process improvements, requirement definition, feasibility, interface testability, maintenance and Reusability.

The Existing system requires a lot of manpower and UI Test Automation time is more. Silverlight is new Microsoft technology and there is no proper Software for UI Test Automation. This system is developed in such a way that the testers can do UI Test Automation in user – friendly manner.

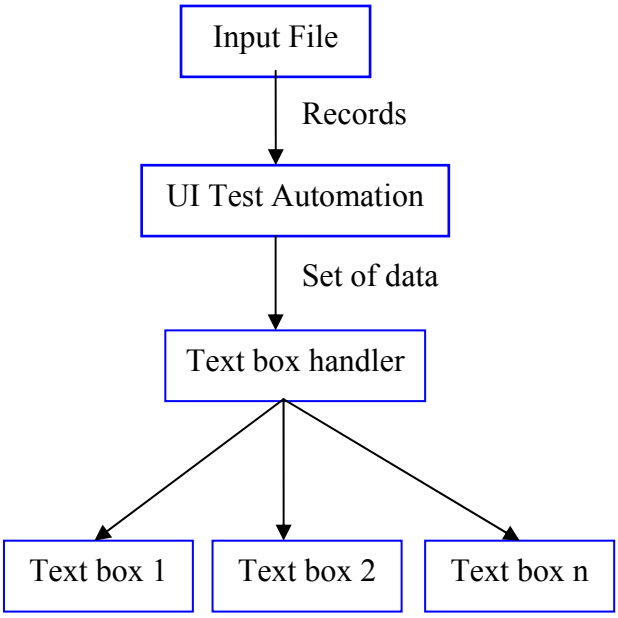

Figure1. UI Test Automation Data flow for text box.

Advantages of User Interface Test Automation are:

- Improved visualization by Micro Soft Dot Net Silverlight
- Online UI Test Automation
- Dot Net Silverlight Applications User Interface Testing
- Testing User Interface with Business logic
- More Secured page User Interface Testing
- Improves the quality of the Applications
- Manual testing is reduced
- Permutation and combinational test cases testing and automation
- Cost and Time is reduced
- Three layers(UI/Business/Data) testing and automation
- Improves and helps in testing
- Emerging new technology testing and automation
- Accessibility-Testing-Automation-Imputation
- Checks all possible test cases
- Convert data from XML/Excel To Database Through User Interface
- User Interface Testing and Automation

IV. Agent Technology

Agents are autonomous, computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors. To say that they are autonomous means that to some extent they have control over their behavior and can act without the intervention of humans and other systems.

Intelligent indicates that the agents pursue their goals and execute their tasks such that they optimize some given performance measures. It means that they operate flexibly and rationally in a variety of environmental circumstances. Interacting indicates that the agents may be affected by other agents or possibly by humans in pursuing their goals and executing their tasks. Interaction can take place indirectly through the environment in which they are embedded.

Advantages of Agents in Multi agent systems are:

- Cooperative agents
- goal oriented
- Decision Support
- Task sharing among agents
- Constrains satisfaction
- Communication and planning among agents
- Coordination and cooperation

The main characteristics of agents in UI Test Automation can be identified as follows:

- System control is distributed to agents
- Data is decentralized
- Data is partitioned and sent to respective agents
- each agent takes care of their data
- Computation is asynchronous

V. Agent Based Ui Test Automation for Silverlight Applications

Any work can be easily done by domain experts and particular agents in real life. Similarly our User Interface Test Automation Agents (UITAA) will take care of UI Test Automation of Silverlight Applications.

The whole UI Test Automation can be split into several modules based on their functionality. They are UI Accessibility, UI Test, UI Automation and UI Test Automation as shown in figure 2.

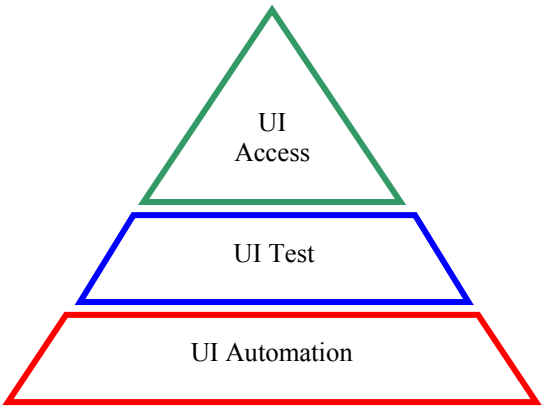

Figure2. Components of UI Test Automation.

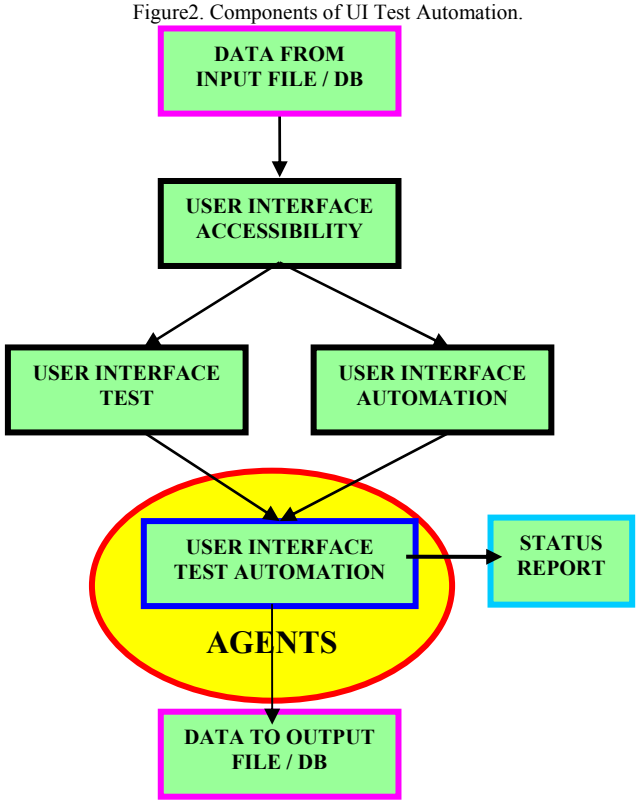

Figure3. Agent based UI Test Automation Data flow.

The data flow of the Agent based UI Test Automation is shown in figure3. UI Test and UI Automation are combined to perform UI Test Automation with the help of multi agent system.

VI. Implementation

Agent based UI Test Automation process for Silverlight applications are implemented as shown in figure4. Our new Agent Based User Interface Test Automation for Silverlight Applications (ABUITASA) gets input record from Excel file. Each record is sent to agents. The agents will split the record and they will send particular data to particular agents to perform UI Test Automation.

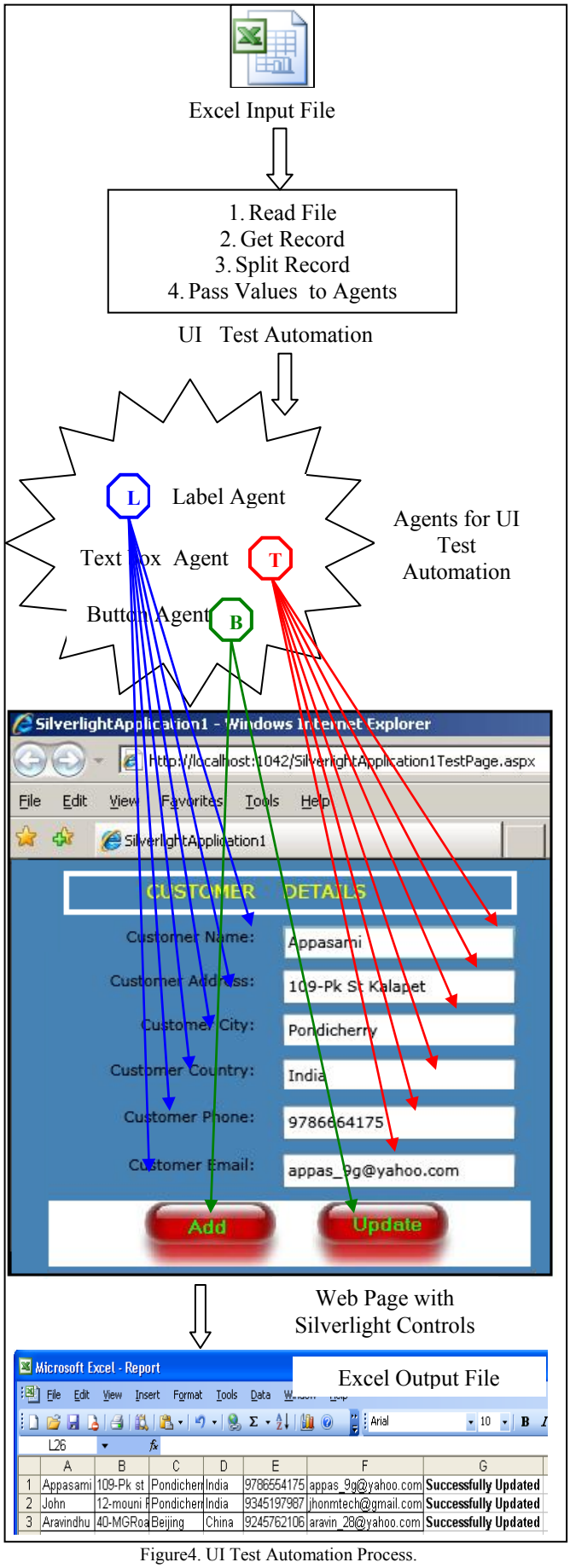

Figure4. UI Test Automation Process.

The Implementation of Agent Based User Interface Test Automation for Silverlight Applications (ABUITASA) is done in .NET 3.5 framework with C#.NET, Silverlight,

XAML, Excel and Visual studio 2008.

The part of the code for Agent Based User Interface Test Automation for Silverlight Applications is given tin figure5. The textbox_agent is used to pass the values to text boxes and the button_agent is used to perform click events.The supplied records are split into data and distributed to particular agents. The textbox_agent is the responsible agent for all text boxes to set the value for text box. Similarly the button-agent is responsible for button actions.

```
public void TestMethod1()
{
Process process =  System.Diagnostics.Process.Get
        ProcessesByName("iexplore").First();
BrowserInstance=        System.Windows.Automation.Auto
mationElement.FromHandle(process.MainWindowHandle);
TreeWalker tw1 = new TreeWalker(new PropertyCondition
(AutomationElement.AutomationIdProperty, "TextBox"));
AutomationElement searchTextBox = tw1.GetFirst
        Child( browserInstance);
TreeWalker tw2 = new TreeWalker(new PropertyCondition
(AutomationElement.AutomationIdProperty, "Button"));
AutomationElement searchTextBox =
        tw2.GetFirstChild(browserInstance);
SilverlightApp app = AvtiveBrowser.silverlightAppas()[0];
app.agent.textbox_agent<TextBox>("TBox1").Text=
        "Appasami";
app.agent.textbox_agent <TextBox>("TBox2").Text=
        "109-pk street Kalapet";
app.agent.textbox_agent
<TextBox>("TBox3").Text="Pondicherry";
app.agent.textbox_agent
<TextBox>("TBox4").Text="India";
app.agent.textbox_agent
<TextBox>("TBox5").Text="9786554175";
app.agent.textbox_agent
<TextBox>("TBox6").Text="appas_9g@yahoo.com" ;
app.agent.button_agent <Button>("ADD").User.Click();
    }
```

Figure 5. Sample code for Agent based UI Test Automation of Silverlight application.

Sample code for Agent based UI Test Automation of Silverlight application is given in figure 5. Initially we have to create one process to get the window handle to control the page. We should create browser handle using window handle to control the window elements. Tree walker entity is created to find a particular element in the web page. Figure5 shows how the values are passed to particular controls through particular agents. For example textbox_agent and button_agent takes care of group of text boxes and buttons.

VII. Conclusion

Agents fill the gap between User Interface Test Automaton and Silverlight applications. Each agent is expert to handle particular type of controls. Agents are simplifies our User Interface Test Automaton. The text box agent is takes care of all text box controls in Silverlight applications to perform User Interface Test Automaton. User Interface Test Automaton plays a vital role in reducing cost, time and man power. It increases the quality of the software.

VIII. Future Work

In future Agents based User Interface Test Automaton for moonlight applications in Linux environment will be developed. User Interface Test Automaton will reduce man power completely in future.

References

[1] Appasami. G and Suresh Joseph. K, "User Interface Accessibility and Test Automation for Silverlight Applications", International Journal of Computational Intelligence Research, Research India publications, Vol. 5, No. 2, 2009.
[2] Appasami. G and Suresh Joseph. K, "Performance analysis of various user interface test automation for Silverlight applications", International Journal of Computer and Electrical Engineering, IACSIT, Vol. 1, No. 4 , pages:475-480, Oct 2009.
[3] Appasami. G and Suresh Joseph. K, "Comparative Analysis of Security and Accessibility of Silverlight XAML with Other User Interface Languages", International Journal of Computer Intelligence Research, Research India publications, Vol. 1, No. 4, pages:490-495, Oct 2009.
[4] Appasami. G and Suresh Joseph. K, "Automation Peer – User Interface Test Automation for Silverlight Applications", IETECH Journal of Advanced Computations,Vol. 3, No. 2 , 2009.
[5] Anna Derezinska and Tomasz Malek, "Experiences in Testing Automation of a Family of Functional- and GUI-similar Programs", International Journal of Computer Science & Applications, Technomathematics Research Foundation, Vol. 4, No. 1, pp.13-26, June 2007.
[6] Q. Xie and A. M. Memon, "Designing and comparing automated test oracles for GUI-based software applications," ACM Transactions on Software Engineering and Methodology, Vol. 16, No. 1, pp. 1-4, Feb 2007.
[7] Zhu Xiaochun, Zhou Bo, Li Juefeng and Gao Qiu, "A test automation solution on GUI functional test", IEEE Conference on Software Maintenance , 6(2): pp: 1413-1418, july 2008.
[8] A. M. Memon, "An event-flow model of GUI-based applications for testing," IEEE conference on Software Testing, Verification and Reliability, Vol. 17, No. 3, pp. 137-157, September 2007.
[9] A. M. Memon, M. E. Pollack, and M. L. Soffa. "Hierarchical GUI test case generation using automated planning". IEEE Transactions on Software Engineering, pages: 144–155, Feb. 2001.
[10] White L, Almezen H. "Generating test cases for GUI responsibilities using complete interaction sequences". Proceedings of the International Symposium on Software Reliability Engineering, 8–11. IEEE Computer Society Press: Piscataway, NJ, 2000; 110–121. October 2000.
[11] White L, Almezen H, Alzeidi N. "User-based testing of GUI sequences and their interaction", Proceedings on Software Reliability Engineering. IEEE Computer Society Press: Piscataway, NJ, 2001; 54–63. 8–11 November 2001.
[12] Memon A, Nagarajan A, Xie Q. "Automating regression testing for evolving GUI software". Journal of Software Maintenance and Evolution: Research and Practice; 17(1):27–64. 2005.
[13] A. M. Memon and Q. Xie. "Studying the fault-detection e.ectiveness of GUI test cases for rapidly evolving software". IEEE Transactions on Software Engineering, 31(10):884–896, 2005.
[14] Q. Xie and A. M. Memon. "Designing and comparing automated test oracles for GUI-based software Applications". ACM Trans on Software Engineering and Methodology, 16(1):4, 2007.
[15] Appasami.G and Suresh Joseph. K, "Automation Peer – User Interface Test Automation for Silverlight Applications", IETECH Journal of Advanced Computations, India, June 2009.
[16] K.P. Sycara and M. Wollridge, "Proceedings of the Second International Conference on Autonomous agents", Association for Computing Machinary, Inc. (ACM), 1998.
[17] Y.Demazeau, "Proceedings of the third International Conference on Multi-Agent Systems", ICMAS-98, IEEE Computer Society, 1998.
[18] Gerhard Wheiss, "Multiagent Systems", the MIT Press,1999.
[19] Fewster, "Software Test Automation", Addison Wesley, 1999.

[20] Kanglin Li and Mengqi Wu, "Effective GUI Test Automation: Developing an Automated GUI Testing Tool ", SYBEX Inc., 2005.
[21] Kanglin Li and Menqi Wu, "Effective Software Test Automation: Developing an Automated Software Testing Tool" ISBN:0782143202 Sybex Inc, 2004
[22] Tom Arnold, Dominic Hopton, Andy Leonard and Mike Frost, "Professional Software Testing with Visual Studio® 2005 Team System", Wiley Publishing, Inc. 2007
[23] Elfriede Dustin,"Effective Software testing", Pearson Education Inc., 2003.
[24] Brad Dayley and Lisa DaNae Dayley, "Silverlight™ 2 Bible", Wiley Publishing, Inc., 2008.
[25] Matthew MacDonald, "Silverlight 2 Visual Essentials" Firstpress, 2008
[26] http://www.silverlight.net
[27] http://code.msdn.microsoft.com/silverlightut
[28] http://silverlight.net/learn/tutorials/controls.aspx
[29] http://www.jeff.wilcox.name/2008/03/silverlight2-unit-testing
[30] http://msdn.microsoft.com/en-us/library/cc645045(VS.95).aspx
[31] http://weblogs.asp.net/scottgu/archive/2008/04/02/unit-testing-with-silverlight.aspx
[32] http://dotnetslackers.com/Patterns_and_Practices/UI_Automation_Testing_with_UIA_Veify.aspx
[33] http://publib.boulder.ibm.com/iseries/v5r1/ic2924/index.htm?info/rzakl/rzaklintroadvantages.htm
[34] http://artoftestinc.blogspot.com/2008/08/automated-testing-of-silverlight.html
[35] http://it.toolbox.com/wiki/index.php/Oracle_11G_New_Features
[36] http://www.adp-gmbh.ch/ora/misc/features.html
[37] http://www.codeguru.com/forum/archive/index.php/t-204260.html