International Journal of Computer and Electrical Engineering, Vol. 1, No. 4, October, 2009
1793-8163

Comparative Analysis of Security and Accessibility of Silverlight XAML with Other User Interface

Appasami.G and Suresh Joseph. K

Abstract - **In this paper we present the security issues in Silverlight. Silverlight XAML is a new GUI (Graphical User Interface) Description Language developed by Microsoft. Now a day's security is the most essential one in web applications. While choosing the web UI languages, we have to consider some parameters like accessibility, Security, Easy development, etc. In this paper we compared XAML languages with other UI languages. XAML Accessibility can be done by accessibility tree. XAML provides more security then other UI languages. Security prohibits Un authorized access and Intrusion. So we have to make our web page as a secured page over internet. Accessibility is mainly used for UI Test Automation. In this paper some popular Graphical User Interface Languages are compared with other UI languages. Using Silverlight XAML we can develop Interactive, attractive and secured web applications. In this paper we present Comparative analysis of accessibility and security of Silverlight XAML and other UI languages.**

Index Terms – **Silverlight, XAML, Graphical User Interface, Accessibility and UI languages.**

I. INTRODUCTION

Silverlight is developed by Microsoft Corporation 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 (Extended Application Markup Language) [15][19][20]. This XAML is a User Interface Description language. In this paper we compared this XAML language with other UI languages. As the Silverlight platform exposes a full accessibility tree trough UIA (User Interface Automation), Test Automation can be easily done using UI Automation [11][14]. We can use UISPY to see the properties of UI controls. In this paper we elaborately discussed the security in Silverlight [17][18].

In this paper we are going to discuss about various GUI Interface Languages and their features. Initially we discussed some important GUI languages [1][7][8], then Silverlight Controls Accessibility for UI test Automation [3][9][10]. Finally we discussed Silverlight security Issues, Security in Silverlight can be achieved by the following ways.

A. Silverlight and the rich client browser

B. From Script-Behind to Code-Behind

C. Secure Code by Design

II. USER INTERFACE MARKUP LANGUAGES FOR A WEB APPLICATION

Ajax, XUL, XAML, Flex, platforms are emerging, each one with its advantages and its drawbacks, and the choice is posed since one wants to build a Web application, a RIA (Rich Internet Application) having the same interface and the same possibilities that a local application as certain popular sites do it. Fear to take the bad way and to spend months under development with an environment on which one could not lead while the other could have proven to be more adapted according post in forums where the question is very present. Things seems even more complicated when one realizes that the various solutions suggested do not cease evolving or moving and proposing functionalities for now. We listed all widely used (for graphical user interface) markup languages, and we have to choose the best one for our needs [2][6][26].

A. AJAX

It is the combination of the technologies of dynamic HTML (CSS, JavaScript, DOM) plus the XMLHttpRequest object which allows asynchronous interaction between the browser and the server. The application uses on the server-side a scripting language like PHP or ASP. Scriptol generates PHP code or binary for mixed applications [30][31].

Advantages:

- Total portability: recognized by all of recent browsers. Very wide choice of frameworks.
- Compatibility with standards of the Web: Javascript, CSS, Document Object Model.
- Implementation is very simple.

Disadvantages:

- The possibilities of client-side interface are limited.
- Do not function locally without a connection to a server.
- Programming in Javascript is very hard.

B. XUL

XUL (XML User Interface Language), pronounced zool, is an user interface markup language developed to support Mozilla, Firefox and Mozilla Thunderbird. XUL reuses many

Manuscript received June 26, 2008.

Appasami. G is with the Department of computer Science, pondicherry University, Pondicherry, India (phone: +91-9786554175).

Suresh Joseph. K is with the Department of computer Science, pondicherry University, Pondicherry, India (phone: +91-9444321492)

existing standards and technologies, including CSS, JavaScript, DTD, RDF and XPCom . It was at first the format of description of interface of Gecko, the rendering engine of Firefox, developed then like a standalone runtime, and distributed under the name of XULRunner. It is possible to install the runtime on any machine and to make a XUL application working locally, but only the Gecko engine recognizes it on the Web. Using XUL is very difficult in particular because of the lack of information. The main benefit of XUL is that it provides a simple and portable definition of common widgets [26][31].

Advantages:
- Compatibility with the standards: Javascript, CSS, RDF.
- Webmasters are not disoriented.
- Integration of HTML code into XUL with HTML tags, in particular forms are easy.

Disadvantages:
- Function only with Firefox and Mozilla browsers.
- Extreme difficulty for programming.

C. XAML

XAML is like XUL an XML-based language of description of interface. But unlike XUL it is at start intended by Microsoft to be a means of creating rich Internet applications. Just as XUL is recognized by Firefox, XAML is recognized by Internet Explorer, with the advantage of a larger number of users. He thus supplements the universal .NET platform, with a language of interface. It is provided natively with Vista and is used for the graphic interface of Vista [32][33].

Advantages:
- Very broad library of functions.
- Choice of the programming language on the .NET platform.

Disadvantages:
- Compatibility limited. Requires practically the presence of WPF to run all the features, and thus requires Vista or Windows XP with the .NET 3.0 extensions.

D. JAVA AND APPLET

The Java solution is complete since it proposes a server-side framework, and with at client-side, on the browser, applets, which are small applications functioning in a Web page. That can also be supplemented by a scripting server-side language, JSP. Let us add to that servlets for Web services and one has a portable and complete platform for enterprise Web applications [34][35].

Advantages:
- Extremely vast library of functions.
- Compatibility with all platforms.

Disadvantages:
- A plugin must be installed on the client-side, and the server must also support.
- Loading of Applets is really slow, which has always limited their use by companies only.
- Programming it is rather complex.

E. Open Laszlo and Flash

Open laszlo is a free application based on a markup language named LZX. On the client-side, it produces Flash code which can be runs thanks to a plugin rather commonly installed, and it produces also DHTML code by Java script [36][37].

Advantages:
- Compatibility for browsers and platforms.
- The force of Laszlo is the existence of a well finalized development tool.
- Many effective users.

Disadvantages:
- A plugin is necessary to let the applications executed.

F. BXML

According to the creator: BXML is short for "Backbase eXtensible Mark-up Language". BXML is a declarative User Interface language, based on XML standards. BXML allows you to develop AJAX applications in a declarative way. This is a commercial product. BXML tags are very similar to HTML tags [38].

G. GladeXML

GladeXML is the XML format used by the Glade Interface Designer. It creates forms that can then be used in conjunction with the libglade library using GTK+. Glade provides a graphical interface development environment in the model of Visual Studio, C++ Builder and so on [39].

H. MXML

MXML is an XML markup language introduced by the Macromedia in 2004. Apart the design of user interface, it can also be used in conjunction with ActionScript to implement complex business logic. alternative from Macromedia to XUL and XAML is used by Web applications in conjunction with developments dedicated for the interaction with the browser. It has a development tool named Flex. [40].

I. UIML

According to the website, the goal of UIML is to create an open standard user interface description language in XML that can be freely implemented by anyone. The motivation is to facilitate better tools for creation of user interfaces that work on any platform available today, but which also will allow today's legacy user interfaces to evolve to new forms for use on platforms that are created years from now [41][42].

J. XForms

The XForms standard has been defined by W3C to combine XML and forms on the web. The standard is intended to be more general, and allows input of data from within desktop applications. It replaces in XHTML the form system used for now in HTML [43]. It is made of three parts:

1) The XForms User Interface provides controls that are targeted toward replacement of HTML's form controls.
2) XForms also defines XML instance data, a structured XML format for data collected by XForms controls.
3) A third part, XForms Submit Protocol, defines how data are sent and received.

K. SVG

The markup vectorial drawing **SVG** (Scalable Vector Graphics) format makes it possible to build graphic components for an interface of Web application. However a framework is needed for that, and there is no emerging solution for now in particular in the world of free software. More again, the SVG recognition by browsers is imperfect. SVG is useful for scalable graphics [44].

Table 1 shows the comparison of most popular UI Languages with Accessibility and security level.

	Year	Development	Runtime	Processing	Languages	Requirements	Accessibility	Security
GladeXML GNOME	1998	Glade IDE	GTK+	Compiled	C, C++, C#	XML	Easy	Low
XUL Mozilla	1998	Text editor	XULRunner	Interpreted	ECMAScript, C++	CSS, DTD, RDF, XPath, XPCom	Easy	Low
OpenLaszlo Laszlo Systems	2003	Text editor	Flash Player	Compiled	ECMAScript	CSS, XPath	Medium	Medium
BXML Backbase	2003	Text editor / Eclipse / Visual Studio	BPC AJAX	Interpreted	JavaScript	XML, CSS, XHTML, XPath	Medium	Medium
MXML Macromedia	2004	Flex Builder	Flash Player / Apollo	Compiled	ActionScript	CSS	Medium	Medium
XAML Microsoft	2006	Microsoft Expression Interactive Designer / Editor / Visual Studio	WinFX / Silverlight	Compiled	.NET languages / JavaScript	XPath, .Net	Difficult	High

Table 1. Comparison Table of most popular UI Languages

III. User Interface Accessibility in Silverlight

Accessibility is very important for User Interface Automation (UI Automation) [1][3][4][5]. Silverlight 2.0 was designed to provide accessibility support using the User Interface Automation API [17][18]. This API is available natively on Windows Vista, and can be installed on earlier versions of Windows as part of .Net Framework 3.5. Bridges are available to Microsoft Active Accessibility (MSAA) [20][22], which is an earlier Windows Accessibility API, and to Linux ATK. AT supporting those APIs should work with Silverlight [22][24].

Accessibility has been something of a weak spot in Silverlight 1.0; with only very basic support for alt tags and default actions. It hasn't been all bad, since this led developers to use Silverlight with "Plain Old Semantic HTML" in a progressive enhancement pattern -- which will continue to be useful in some scenarios going forward. But the range of options for accessible development increases substantially with Silverlight 2. Silverlight 1.0 didn't have much in the way of accessibility - essentially the equivalent of alt tags. Silverlight 2 will change that by exposing a full accessibility tree to accessibility tools in builds post Beta 1.

What will Silverlight 2 provide for accessibility?

- Tabbing and tab order
- Focus and Keyboard Input
- Exposed Accessibility Tree (through UI Automation) for screen readers and other accessibility tools
- Accessibility Information directly in XAML Markup via AutomationProperties.
- Support for AutomationPeer as in WPF

If we want to see the accessibility tree for our applications, then we can use UI SPY to see all the properties of UI Controls.

IV. Security of Silverlight Applications

A. Silverlight and the rich client browser

Traditional Web applications are consumed through a special breed of application -- the client browser. Overall, the behavior of the browser hasn't changed much in recent years -- the browser is mostly used to download some content and render it into its client area. When the content is HTML, any embedded JavaScript code will be interpreted and executed. Through JavaScript code, the page author can manipulate elements in the page's Document Object Model (DOM) and trigger remote operations using **XMLHttpRequest**. In other words, JavaScript is the programming language of the browser and it is used to code any task that the page author wants the page to accomplish [13][17][18]. Since the very early days of the Web, two things appeared as clear as facts:

- First, the browser can't just be like a dummy 3270 dumb-and-deaf terminal. It needs be interactive and responsive to some extent.
- Second, enabling downloaded code to run within the user's machine is a security hazard. For this reason, all browsers run JavaScript code in a "sandbox" - a tightly controlled hosting environment that prevents access to critical local resources such as hardware and file system. With a sandbox in place, executing untrusted code embedded in Web pages is generally considered safe.

However, JavaScript is not as powerful as required by modern Web pages. Implementing a good user experience is perhaps far beyond the reach of the JavaScript language -- today and in the foreseeable future. That's why new programming environments are emerging specifically designed to build Rich Internet Applications (RIA). Silverlight 2.0 has is just one of these. Stung by the failure of the ActiveX platform, Microsoft put a lot of attention and effort in the development of Silverlight, especially from a security point of view. Since its first appearance 10 years ago, ActiveX wasn't well received for essentially two reasons -- lack of support for platform interoperability and a poor security model.

Today, Silverlight 2.0 revamps the same core idea of ActiveX - running user-defined and compiled code within the client browser. However, today Silverlight 2.0 does that in a cross-platform way and using an adequate sandbox to protect the user machine. In this article, I examine the new security model of Silverlight 2.0. It Shipped with a cross-platform (and obviously thinned down) version of the .NET Framework, Silverlight 2.0 lets developers build rich managed applications that run in virtually any client browsers [21][23]. As a developer, you can write Silverlight applications using a variety of .NET languages, including C#, Visual Basic, Managed JScript, and also dynamic languages such as IronPython and IronRuby. Figure 1 provides an overview of the Silverlight architecture and breaks the supported .NET Framework into pieces [12][13][15][16].

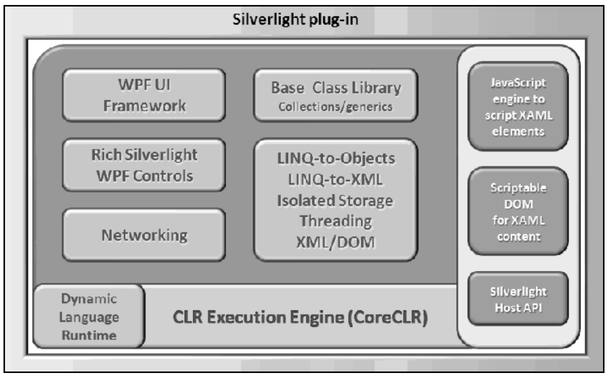
Silverlight plug-in
WPF UI Framework
Base Class Library Collections/generics
Rich Silverlight WPF Controls
LINQ-to-Objects LINQ-to-XML Isolated Storage Threading XML/DOM
Networking
JavaScript engine to script XAML elements
Scriptable DOM for XAML content
Silverlight Host API
Dynamic Language Runtime
CLR Execution Engine (CoreCLR)

Figure 1: Architecture of the Silverlight 2.0 engine.

The Silverlight 2.0 plug-in is made of a core CLR environment and a brand new Dynamic Language Runtime (DLR) environment made to measure to process dynamic languages such as Python. (IronPython is a Microsoft .NET language with a Python-compatible syntax.) Any code that runs within the CoreCLR may target classes in the Silverlight supported subset of the .NET Framework [16][17]. This subset includes collections, generics, as well as more specific application programming interfaces such as that for XML parsing, isolated storage, LINQ-to-Objects. Each instance of the Silverlight plug-in is bound to a URL that returns XAML with optional code-behind classes written in any supported language. In Silverlight 2.0, the XAML supports a compatible subset of the full WPF platform along with some specific common controls not available (yet) to the desktop platform. It is essential to note that Silverlight 2.0 does not require any version of the full .NET Framework to be installed on the client machine. The Silverlight setup program downloads everything that is necessary to enable all the features in Figure 1 on a standard Windows, Mac OSX, or Linux machine. For more information about running Silverlight on Linux, see the Moonlight project, the plug-in's code-name for Linux. Currently, Moonlight is under development.

The Silverlight plug-in measures about 4 MB in size and normally won't take more than just a few seconds to install on a machine [19][21][23]. You need to install it only once and you can then successfully navigate to any Web page that hosts Silverlight content using your browser of choice. Silverlight applications can be deployed to any Web server including Apache on Linux, and always require some sort of host page. You can't just serve plain XAML data to the browser. Instead, you need a host page that sets up the plug-in and make it point to the URL for the XAML content. The host page can be a static HTML files or any server-side generated page: ASP.NET, ASP.NET AJAX, PHP, Java, Python, Ruby, and so forth.

B. *From Script-Behind to Code-Behind*

Any elements that appear in a XAML document can be scripted by the companion code that comes with the XAML resource. In Silverlight 1.0, XAML elements can only be scripted using JavaScript functions and their events can be handled exclusively through JavaScript handlers. A common naming convention entails that you name the JavaScript file that contains this code after the page that hosts the plug-in. For example, a page named default.aspx that incorporates Silverlight will be served by a script file named default.aspx.js. This is only a convention, however, and is sometimes referred to as "script-behind". You typically create XAML documents using the new facilities in Visual Studio 2008 or Expression Studio tools. You then empower these documents using C# or Visual Basic code saved to a classic code-behind file, such as default.aspx.cs. Any code-behind class attached to a XAML document will be downloaded to the client and executed within the local machine [19][20][21]. Here's how a XAML document links to a managed class:

```
<UserControl x:Class="Samples.MyPage"
  xmlns="http://schemas.microsoft.com/client/2007"
  mlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Width="400" Height="300">
  <Grid>    :  </Grid>
</UserControl>
```

The **x:Class** attribute on the **UserControl** XAML tag instructs the plug-in about the class to instantiate to start up the application. **UserControl** is generally the root tag of a Silverlight 2.0 page. Where does class **Samples.MyPage** come from? It is defined in the code-behind file of the XAML document and inherits from **UserControl**, as in the code snippet below:

```
namespace Samples
{   public partial class MyPage : UserControl
  {  public MyPage()
     {  InitializeComponent();
     }
     // Event handlers and helpers go here
  }
}
```

The **x:Class** attribute is the discriminating factor that enables the CoreCLR to run managed code. An XAML document that lacks the attribute can only use JavaScript to handle internal events and provide glue code. If you use Visual Studio to develop your Silverlight 2.0 application, XAML markup and any related code are compiled into a standard .NET assembly and then packaged in a XAP file along with any required auxiliary resources such as images or perhaps script files. Additional assemblies, if necessary, are added to the XAP bundle as well. What would be the typical size of these packages? A XAP file made of a code-behind class with no external dependencies will hardly exceed 10 KB and often it is around the base size of a .NET assembly, which is 4 KB. If your code depends on external assemblies, all of them are added to the XAP [20][21].

C. Secure Code by Design

In the .NET Framework, upon loading an assembly the CLR gets evidence for it and figures out the code group of the assembly. A code group defines the list of privileged actions that assemblies are allowed to perform. Whenever some code is about to execute a privileged action, the CLR verifies its permissions and throws if it finds out anything wrong. Permissions for code groups are determined by the machine administrator. The administrator is responsible for defining the security policy for a given machine. In theory, the administrator can also disable code access security altogether. This model of handling security is known as "code access security" (CAS). At its core, with CAS enabled any method of a .NET Framework class can execute any action unless the assembly it belongs to lacks permissions for it. The Silverlight 2.0 implementation of the CLR (also known as CoreCLR) reverses this principle completely. Any code that goes through the CoreCLR is considered partial trust and is not allowed to call into methods that require higher privileges. The CAS model is not supported in Silverlight 2.0 [19][21]. Code access security in Silverlight 2.0 is guaranteed by a brand new security model. Security in CoreCLR is based on attributes, as in Table 2.

Security Attribute	Description
Security Transparent	Any code marked with this attribute runs as partial trust and is not allowed to perform any calls that would elevate the privileges of the call stack. *The attribute is supported since version 2.0 of the .NET Framework.*
Security SafeCritical	Any code marked with this attribute runs as full-trust and may be called by transparent code. *The attribute is new to Silverlight 2.0.*
Security Critical	Any code marked with this attribute will always run as full-trust. *The attribute is supported since version 2.0 of the .NET Framework.*

Table 2: Silverlight 2.0 security attributes.

Each attribute identifies a different level of security hazard associated with the code. The concept of security transparent code is nothing new in the .NET Framework. It refers to code that is marked as unable to call into full-trust code and perform any other action that would possibly elevate the permissions of the call stack. Within the fully-fledged CLR, transparent is only the code that belongs to assemblies explicitly decorated with the SecurityTransparent attribute. Likewise, transparent is also any method or class that is explicitly given the attribute, as below.

```
[SecurityCritical(SecurityCriticalScope.Everything)]
public class Foo
{    public void DoSomething()
    {
      :
    }
}
```

Unlike the fully-fledged CLR, in Silverlight 2.0 the CoreCLR considers any code as transparent unless it is decorated with a different security attribute. In particular, this means that all the code in a Silverlight 2.0 application cannot directly execute any critical operations such as invoking unsafe or unverifiable code or attempting system-wide changes through the P/Invoke subsystem. Application code in the Silverlight 2.0 virtual machine runs as partial trust and can only invoke other transparent code or, at most, code marked with the **SafeCritical** attribute. What kind of animal is a safe-critical method, class, or assembly? As in Table 1, this attribute doesn't exist in the .NET Framework and has been added just for implementing the Silverlight's security model. The **SafeCritical** attribute identifies code that sits in between the application code (what users download from the Internet) and the platform code (where some critical methods are defined) [19][21][22].

XAML is very popular and more secured than other UI languages, because Silverlight XAML is designed as open standard and developed by Dot Net Frame work 3.0 by

Microsoft [19][20].

V. Conclusion

Silverlight UI Controls are Interactive and Attractive. Since it provides more security, it is better than other UI Languages. Silverlight 2.0 enables .NET developers to extend their existing skills and investments to build cross-platform rich Internet applications that work in a cross-platform manner and are secure at the root. This paper presents security implementation in Silverlight 2.0 applications. XAML User Interface Silverlight language is better than other UI languages. Using Accessibility

VI. Future Work

In future, a common open standard Graphical User Interface Language will be developed. It will rectify the pitfalls of the available User Interface languages. It will be very fast in rendering and Provides more security.

References

[1] 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. 4-es, February 2007.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Q. Xie and A. M. Memon. “Designing and comparing automated test oracles for GUI-based software Applications”. ACM Transactions on Software Engineering and Methodology, 16(1):4, 2007.

[11] X. Yuan and A. M. Memon. “Using GUI run-time state as feedback to generate test cases”. In ICSE’07, Proceedings of the 29th International Conference on Software Engineering, pages 396–405, Minneapolis, MN, USA, May 23–25, 2007.

[12] Fewster, “Software Test Automation”, Addison Wesley, 1999.

[13] Kanglin Li and Mengqi Wu, “Effective GUI Test Automation: Developing an Automated GUI Testing Tool ”, SYBEX Inc., 2005.

[14] Kanglin Li and Menqi Wu, “Effective Software Test Automation: Developing an Automated Software Testing Tool” ISBN:0782143202 Sybex Inc, 2004

[15] Tom Arnold, Dominic Hopton, Andy Leonard and Mike Frost, “Professional Software Testing with Visual Studio® 2005 Team System”, Wiley Publishing, Inc. 2007

[16] Elfriede Dustin,“Effective Software testing”, Pearson Education Inc., 2003.

[17] Brad Dayley and Lisa DaNae Dayley, “Silverlight™ 2 Bible”, Wiley Publishing, Inc., 2008.

[18] Matthew MacDonald, “Silverlight 2 Visual Essentials” Firstpress, 2008

[19] http://www.silverlight.net

[20] http://code.msdn.microsoft.com/silverlightut

[21] http://silverlight.net/learn/tutorials/controls.aspx

[22] http://www.jeff.wilcox.name/2008/03/silverlight2-unit-testing/

[23] http://msdn.microsoft.com/en-us/library/cc645045(VS.95).aspx

[24] http://weblogs.asp.net/scottgu/archive/2008/04/02/unit-testing-with-silverlight.aspx

[25] http://dotnetslackers.com/Patterns_and_Practices/UI_Automation_Testing_with_UIA_Veify.aspx

[26] http://www.xul.fr/comparison-user-interface-markup-languages.html

[27] http://www.scriptol.com/ajax/ajax-xul-xaml.php

[28] http://www.ddj.com/windows/206902613

[29] http://msdn.microsoft.com/en-us/library/aa302426.aspx - Secured web controls

[30] http://www.ajaxtutorial.net/

[31] http://www.xul.fr/en-xml-ajax.html

[32] http://msdn.microsoft.com/en-us/library/ms752059.aspx – XAML

[33] http://www.devx.com/webdev/Article/20834 - XAML

[34] http://www.java.sun.com/applets/ - Applets

[35] http://www.appletcollection.com/ - Applets

[36] http://www.openlaszlo.org – open Laszlo

[37] http://www.aspnetflash.com/ - Flash

[38] http://www.alexa.com/siteinfo/bxml.net - BXML

[39] http://www.library.gnome.org/devel/libglade/unstable/Gladexml.htm

[40] http://www.adobe.com/.../coding_with_mxml_and_actionscript/

[41] http://www.uiml.org

[42] http://www.oasis-open.org/committees/uiml/

[43] http://www.xml.com/pub/a/2001/09/05/xforms.html

[44] http://www.adobe.com/svg/

Appasami. G was born in Pondicherry, India in 1980. He received his Master of Science degree in Mathematics and Master of Computer Applications degree from Pondicherry University, Pondicherry, India; Pursuing Master of Technology in Computer Science and Engineering from Department of Computer Science, Pondicherry University, Pondicherry, India.

Currently he is working as Research Assistant in Department of Computer Science, Pondicherry University, Pondicherry, India. His Area of interests includes image processing, neural network and web technology.

Suresh Joseph. K was born in Tamil Nadu, India in 1978 received his Bachelor of Engineering degree from Bharathiyar University and Master of Engineering (Computer Science and Engineering), from University of Madras, Chennai, India, in 2002.
Since 2006, he has been an Assistant Professor at the Department of Computer Science, Pondicherry University, Pondicherry, India. His research interests include Image processing and Softwar