Research on RIA frameworks and cross platform RIA porting between Flex/Flash and Silverlight/Microsoft .NET

BTECH 450
End of Year Project Report

Jason Wei-Lun Hsia
Department of Computer Science
University of Auckland

Supervisors

Academic Mentor:
Dr. Xin Feng Ye
University of Auckland
New Zealand

Industry Mentor:
Chris Skilton
Futuretech
New Zealand

Department of Computer Science, University of Auckland, Auckland, New Zealand

jhsi014@aucklanduni.ac.nz

Abstract. The topic of my Bachelor of Technology final year project is on researching on the two most popular Rich Internet Application (RIA) frameworks/languages—Adobe Flex and Microsoft Silverlight. This report looks into the different aspects of the two RIA frameworks, including portability, performance, security, usability, and discusses on the similarities and differences between the two frameworks. As a result of these investigations, development guidelines and a basic prototype of Flex to Silverlight code porting auto-generator was developed using a compiler tool called ANTLR.

Keywords: Adobe, Flex, Flash, Microsoft, Silverlight, RIA, Rich Internet Application, porting

Table of Contents

1 Introduction

This report documents the information about the BTech 450 project that I am currently doing for a company called Futuretech. The project is to research on the methodologies of porting application from Flex to Silverlight, and the different aspects of the RIA frameworks.

1.1 About Company

The company that is supporting me in this project is called Futuretech. Craig Meek founded Futuretech in 2008 and is a specialist in visual communication. The company specializes in data visualization, and has been working closely with HP R&D programs to produce visual business intelligence products, which are all based on the Adobe Flex framework.

1.2 About RIA

Rich Internet Applications (RIA) is the modern trend in Internet programming. It has gradually becoming the mainstream within businesses, and slowly replacing the traditional HTML web pages as they have the advantage of being more attractive, viable, and provide developers a richer range of controls, functions, and features to work with.

Rich Internet Applications are web applications that have many of the characteristics of desktop applications. They allow programmers to develop applications that are cross platform, as they are supported through Internet browser plug-ins or virtual machine environments.

For many years the predominant RIA framework has been Adobe's Flash engine. This can arguably be primarily attributed to the vast acceptance and proliferation of the Flash Player plug-in. Microsoft joined the RIA battle in 2007 with a RIA platform called Silverlight. Since Silverlight v1 was released, Microsoft have continued development of the platform releasing version two, the incumbent version three, and are poised to release version four final in coming months. They have packaged the Silverlight plug-in with the latest versions of Internet Explorer and have made a commitment to establishing versions of the plug-in for Mac, Linux, and mobile platforms including Symbian. Microsoft are

rapidly becoming a serious contender for RIA dominance, heading for a showdown of sorts with Flash, and are perhaps politically advantaged within Enterprise thanks to a history business orientated support products, tools, and applications.

The emergence of Silverlight as a genuine threat to Flash, and Adobes flagship RIA framework Flex, has given those specializing in RIA development much to think about. Namely, should suppliers of RIA technology be reconsidering their choice of platform, and if they are to make the switch, how can they efficiently port their Intellectual Property into an alternative framework, whilst still maintaining core application architecture? In addition to this consideration, there is likely to be a period of time within which each RIA platform choice offers advantages over the other, and navigating through this period of turbulence will require a clear strategy and keen. During this period it may be advantageous for RIA developers to consider bridging between Silverlight and Flex, maintaining two distinct code bases simultaneously.

1.3 Motivation

Over the past few years Microsoft had been pushing the development of Silverlight, hoping that it catch up with the more popular RIA framework—Adobe Flex. (Just over 3 years from their first release, Microsoft has just released Silverlight 4) In many aspects, the Silverlight has achieved that goal; it provides similar functionalities to those in Flex (Flash), such as integrating multimedia, graphics, animations, and interactivity features.

Due to the rapid growth of the Silverlight framework, there has been an increase in market demand for Silverlight applications. Thus, it really has caught companies' eyes to make a move or looking into the possible transitions from other RIA framework developments into Silverlight developments. There has been an increase in market demand for Silverlight applications. The company has getting more and more enquiries about the Silverlight-based products from their clients.

Thus the company would like to be able to some kind of documentation or guideline to enable them to create the building blocks of a Silverlight powered equivalent of their Flex based applications. They have a real-world requirement for such and suspect that there are many companies with investment in one platform or the other that would like to mitigate risk of commitment also.

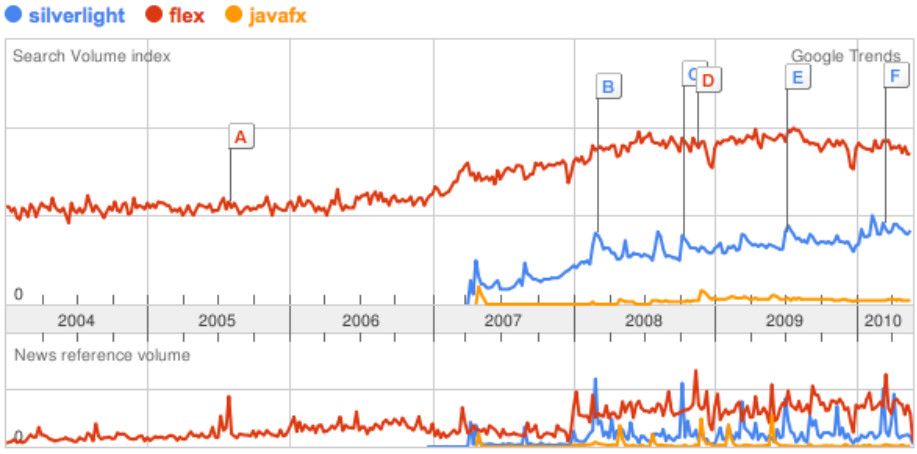


Figure 1 Google Trend analysis on the 3 most popular RIA platforms

1.4 Project goal

The main goal of the project is to do research on Rich Internet Application (RIA) frameworks/languages—Adobe Flex and Microsoft Silverlight especially. From the information gathered, then define a set of processes and policies to enable the company to create the basis of a Silverlight version of the company Flex based software applications. It outlines a methodology and process for synchronous code base porting of MXML to XAML, AS3 to C#, Java to C#, AMF to SOAP, etc. In addition it discusses the best graphic design applications for design assets to support both RIA frameworks.

Since the RIA market has been dominated by the Flash/Flex framework in the past, most of the RIA software development companies have based all their development in Flex. So as the ultimate goal of this project, in terms of helping the business, is to port the Flex based source codes into the equivalent Silverlight version codes.

There is no guarantee that either RIA framework will greatly outshine or outlast the other; in actual fact many insiders also consider the development of HTML 5 and CSS 3 to break RIA out of its plug-in sandbox. Many opinions reside, but the company's collective opinion focuses on the likelihood of Silverlight v4 giving them good reason to port their applications from the Flex/Flash 4 SDK, maintaining parallel development in both platforms without having to duplicate 100% of their effort.

From the academic point of view, it will be focused on comparing and contrasting the features of the two Rich Internet Application (RIA)

frameworks—Flex and Silverlight. These include the usability, portability, performance, and security.

As the ultimate goal of this project, we would like to produce an auto code generator that will take the Flex/Flash source code and convert it into an equivalent version in Silverlight code.

2 Technologies & Development Tools

The main focus of technology in this project is obviously the Rich Internet Application (RIA). The project will investigate the two RIA frameworks—Flex and Silverlight.

2.1 Rich Internet Application (RIA)

Most RIA frameworks use the Model-View-Controller (MVC) as their software architectural where the applications developed under these frameworks are broken down into separate layers—data model, user interface (view), and the business logic (controller). In this architecture, the model is responsible for the gathering of data, the view presents the data from the graphically and receives user inputs, and the controller is responsible for processing the user request from the UI controls, and updates the model accordingly.

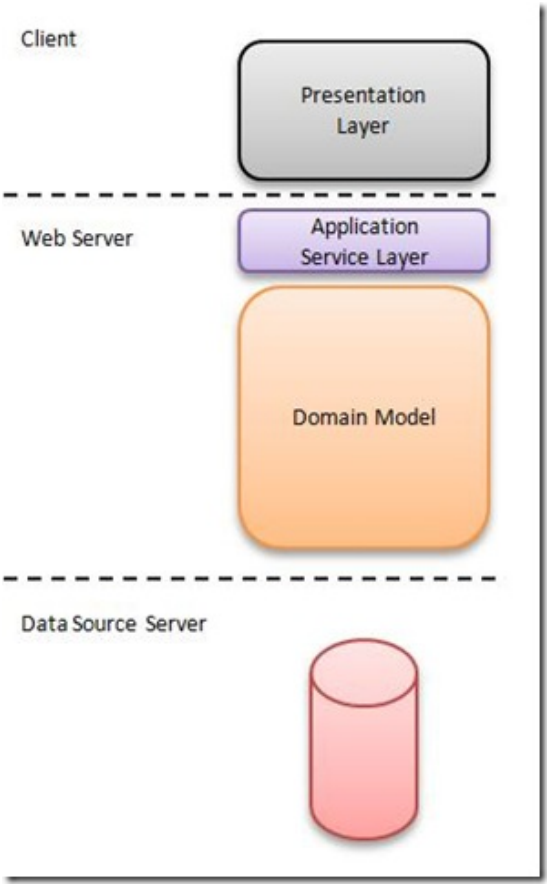


Figure 2 Typical RIA architecture

2.2 Adobe Flex

The research work I have done on Flex is looking into the basic data types, user interface controls, and the functionalities. Flex is one of the earliest RIA platforms, and remains the most popular RIA platform today. It uses MXML and ActionScript

Since the update from Flex 3 to Flex 4, there has been a new framework in place for Flex on top of the existing **MX** framework—**Spark** Framework. The advantage of the spark framework over the old MX framework is the separation of visual controls and the application logic.

2.3 Difference between Flex and Flash

There seems to be a misconception that people have with Flex and Flash. Flex is not a new language. One of the main differences is the use of MXML, which is markup file for defining an application's UI controls and some script code. However, the truth is that the MXML file gets compiled back into ActionScript file anyway, and most of the business logic and functions of a Flex program is defined and implemented in ActionScript. It uses "ActionScript just like flash does, but it comes with a lot of extra features, to make more intelligent applications". Overall, developments that are more focused on the animation and design are likely to do their development using Flash, whereas business application that requires database accesses, implementations of business logic are more likely to use Flex development environment.

2.4 Microsoft Silverlight

The research work I have done on Flex is looking into the basic data types, user interface controls, and the functionalities. Silverlight basically uses the Windows Presentation Foundation (WPF), but in a smaller package version, and is missing some features and controls that are found in the WPF framework. It only uses a faction of the entire .NET Base Class Library. However, that is essentially what Silverlight aims to do—a lightweight, cross-platform and a lightweight client.

2.5 Difference between Silverlight and WPF

As with Flex and Flash, many have mistaken that Silverlight is complete new technology. Whereas, Silverlight is really just a subset of Windows Presentation Foundation (WPF), the latter is used for building desktop applications using the .NET framework. WPF framework contains whole sets of Windows UI components that can be used in the Silverlight as well. This really is a bonus for developers who have experiences in .NET programming, as they should be able to adapt and develop Silverlight applications fairly easily. But that raises another problem, since Silverlight basically uses the same development environment as WPF, as development tools such as Visual Studio and Expression Blend are all windows-based only software. This means that for non-windows users, they are basically out of the game, such as Linux Ubuntu and Mac OS X, largely limits the popularity it can generate if it is to support all platforms.

2.6 Similarities and Differences between Silverlight and Flex

Adobe and Microsoft have each released useful and popular IDE tools to develop application in their own RIA frameworks. In the Silverlight development environment there are two main development tools (IDE) that most people use to build Silverlight applications—Microsoft Visual Studio and Microsoft Blend. Then there is Adobe Flex Builder to develop Flex framework applications.

The two RIA frameworks have a lot of features and function in common. Mainly because Flex has been dominating the RIA market before Silverlight was released (Flex is still the most popularly used RIA platform, but Silverlight is slowly but gradually increasing in the market), thus it became necessary for Silverlight to replicate the same features and architecture of Flex.

Source File architecture

Flash	Silverlight
ActionScript	C#
MXML	XAML
FLA, Flex Project	.NET Project
.SWF	.XAP
Flash Professional	Expression Blend
Flex/Flash Builder	Visual Studio

Figure 3 Comparison table for the two RIA frameworks on the different RIA components

In the figure above, it shows the equivalent components in RIA between the two frameworks. First of all, both frameworks use a markup language to visually represent data structures that will be parsed and translated into code. For Silverlight it is XAML files and MXML for Flash.

According to the research and development I have done, XAML seems to be the more complex markup language compared with MXML—it contains more details, and well designed, but might be harder to use and takes longer to learn compared with MXML

```
55              {
56                  employees = event.result.employees.employee;
57              }
58
59
60              protected function employeeService_faultHandler(event:FaultEvent):void
61              {
62                  Alert.show(event.fault.faultString, "Fault Information");
63              }
64
65          ]]>
66      </fx:Script>
67
68      <!-- Declarations ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
69
70      <fx:Declarations>
71
72          <s:HTTPService id="employeeService"
73              url="http://www.adobetes.com/f4iaw100/remoteData/employeeData.cfm"
74              result="employeeService_resultHandler(event)"
75              fault="employeeService_faultHandler(event)" />
76
77      </fx:Declarations>
78
79      <!-- UI components ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
80
81      <s:Label text="Company Vehicle Request Form"
82          x="80" y="34"
83          styleName="addHeader"/>
```

Figure 4 MXML code with ActionScript

Development Environments/Tools

Lets start off talking about Silverlight, from my own programming experiences, Silverlight development is really easy to grasp if you are familiar with the C# programming and the Microsoft .NET framework. As mentioned before in the report, Silverlight uses a scaled down version of the .NET framework, and it uses C# to implement the business logics behind the user interface controls. (May be used to program entire applications including UI)

2.7 Advantages & Disadvantages

Both frameworks has got their own advantages and disadvantages, it is still unclear which the clear winner is. First of all, Flex has the advantage of having high market share, as it has been around basically ever since RIA is used, almost every computer in the world has got Flash player plug-in installed [2] (95% to be exact). Also, Flex development tools are supported in both Mac OS and Windows, whereas Visual Studio and Blend (Silverlight's IDE tools) are only available in Windows OS.

Another advantage of Flex, again due to its popularity over the past years, it has a large range of IDE development tools available—Flash Development Tool, ActionScript Development Tool, FlashDevelop, SE|PY, ANT builds, and more. However, according to many articles and in terms of tool functionalities, Microsoft Blend seems to have an edge over the Flash Catalyst, in designing and

constructing RIA. Blend is mainly used in Silverlight to build its markup structure of the RIA UI components in XAML format, and Visual Studio is responsible for business logic coding in C# and debugging the applications.

From a developer tool point of view, Flex has its advantages as the code hinting (in Adobe Flash Builder) is done nicely and is quite handy for both beginners and expert users. On the other side, Microsoft's version of such feature is Intellisense, but the support and help is gives is not as direct and helpful as the one in Flash Builder. However, one could argue that this is due to the large and complex base code library that the .NET framework has, thus making choosing the correct code for user can get quite complicated compared to the smaller Flex framework.

Being a latecomer as Silverlight is, it does have some of its own unique features compared with Flex. Silverlight supports multithreading natively which will 'free up' the user interface and putting work evenly across multiple threads/processors. Another feature is the use of LINQ; Silverlight uses the LINQ query language just like any other .NET developments. Other useful features include deep zooming, runtime XAML parsing, and behavior triggers (allows programmers to use generic actions triggered by events)

2.8 Learning Resources

At the beginning of this project, I basically have no knowledge or any experience in Flash or Flex development. Although, I have some experiences in the Microsoft .NET framework and C# programming, I have no knowledge and experience in the Silverlight architecture and development. Thus, it was quite a steep learning curve for me. Below, I will mention about the resources that have used in my research and learning.

Adobe Flex

The material that I have used to get familiar with the Flex development environment is the online Adobe developer Connection video tutorials and web page materials. For Flex examples or other information I used the Adobe Tour de Flex desktop application, which was suggested by the company that I'm working with.

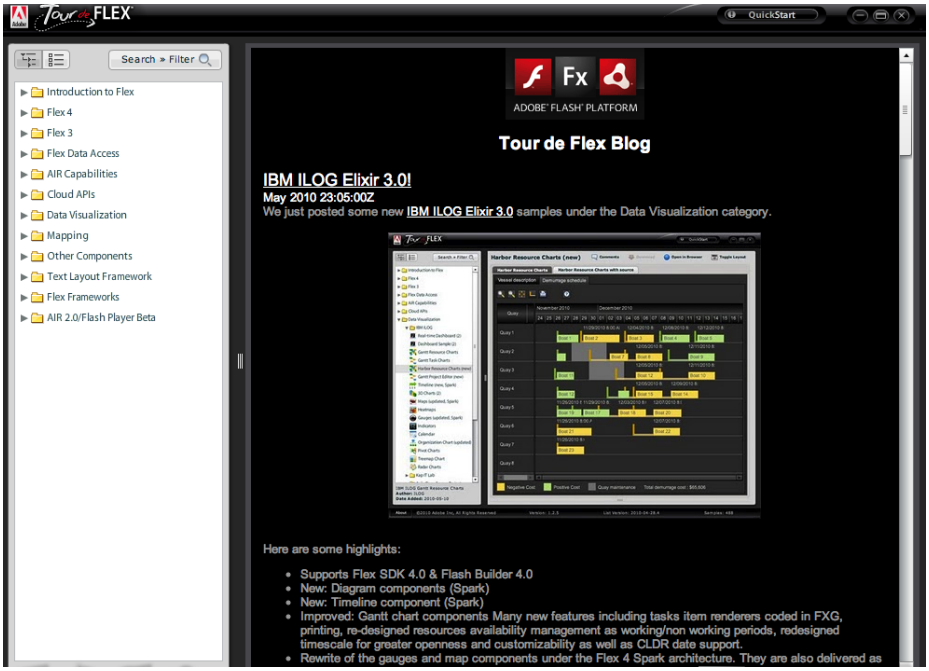


Figure 5 Tour de Flex application

Microsoft Silverlight

In terms of Silverlight, it really was hard to find a complete learning material to learn the framework in detail. I started off using the learning materials on the official Silverlight website, but it wasn't done it a structured manner. However, luckily I found out that one of the University of Auckland courses was teaching Silverlight programming from scratch—INFOSYS320. Thus, I took some of that course's material off someone who was taking that course, and started learning Silverlight in detail from the beginning again.

3 Guideline and policies

One of the main goals for this project is to define a set of guidelines and policies to enable the company to create the building blocks of a Silverlight version equivalent of the company's Flex based applications. Here, I have tried out numerous basic examples, and research on the two frameworks and came up with the following guidelines for Flex developers to construct their code in such a way that is similar to Silverlight's application structure, thus making it easy to convert the code into Silverlight.

3.1 Code Structure

In Silverlight, the XAML markup file defines the visual appearance of an application's user interface, and is associated with the code-behind C# file defining the business and program logic. The UI design can be adjusted without necessarily making changes to the logic in code-behind. XAML in this role simplifies the workflow between individuals who might have a primary visual design responsibility and individuals who are responsible for application logic and information design. [3]

One important thing about Silverlight application is that there is no script tag inside the UI markup XAML file, meaning that all code must written in the code-behind files, C#. Thus, it is important for Flex developer to keep their MXML script-code free (all code goes into code-behind ActionScript file), to make the code porting to Silverlight much easier.

3.2 Component Styling

Styling of UI control components are done very differently between the two frameworks. In Flex, it takes advantage of the Cascading Style Sheets (CSS) used in HTML styling. However, Silverlight does not use CSS technology, instead styling definitions is done within the XAML file in the form of property setters. What they both in common is that the styling can also be done inline, however if you are to reuse the same style settings later on in the application you need to store the property setters are separate resources.

At first, styling in Silverlight might seem more difficult as it does not use a CSS file to define, however the Microsoft Blend tool will help you define the style visually, thus making the task much easier. Once the style editing is completed in Blend, it will store the settings as a resource so developer will be able to reuse this style implementation. A later declared component will be able to reuse the same style be inheriting the same style. This can be done by using the 'BaseOn' property [4]. (Shown below)

```
<Style x:Key="BaseButtonStyle" TargetType="Button">
        <Setter Property="FontWeight" Value="Bold"/>
        <Setter Property="Background" Value="#FF49A2ED"/>
        <Setter Property="Cursor" Value="Hand"/>
</Style>
<Style x:Key="ExtendedButtonStyle" TargetType="Button"
BasedOn="{StaticResource BaseButtonStyle}">
        <Setter Property="Background" Value="#FFFC212E"/>
</Style>
```

In Silverlight, instead of the CSS file acting as the style template, the same type of style template can be defined within the XAML file as well, and using the same 'BaseOn' property as shown above to make the reference.

```
<ControlTemplate x:Key="ExampleTemplate" TargetType="Button">
…
</ControlTemplate>


<Button Padding="10" VerticalAlignment="Center" Content="Button"
HorizontalAlignment="Center" Foreground="White"
Template="{StaticResource ExampleTemplate }"/>
```

From this we can see that it is good practice for Flex develops to write all its styling in CSS files, which can converted nicely into the XAML control templates in Silverlight XAML file. Whereas defining components in-line with the UI tags will make the conversion to Silverlight code a disaster job.

3.3 Animation

Animation is one area that Flex and Silverlight are fundamental implemented differently with each other, thus making it almost impossible to convert straightforwardly. Silverlight supports animation through a time based model and not a frame-based [5]. Known as WPF animation model, it has entirely eliminated matrixes. The system figures out the function just with the defined start and end conditions. Although, there is no real guidelines can be given for animation, but animation implementation in Silverlight seems far easier than Flex/Flash where calculation of matrices is required to work out the elements in each frame.

3.4 Data Binding

Another area that the two frameworks are vastly different in is data binding. In Flex, most of linking and connections with events and property changes are done by the code generated by the compiler user merely have to insert the [Bindable] Meta tags to the right components. Whereas in Silverlight, all these hard works are required to be implemented by the developers, such as implementing the 'INotifyPropertyChanged' interface to handle change notifications, and dependency property. All these mean that Silverlight has a more rigid architecture when it comes to data binding, and Flex provides developers with more flexibility. As a guideline for Flex developers, there is simply more work to do in order to construct bindings in Silverlight. Few things to remember, in Silverlight you need to define access modifiers for all the elements that you wish to use data binding with.

```
public string FirstName
{
     get { return _firstName; }
     set { _firstName = value; }
}
```

Also, all data binding in Silverlight requires both a source object and property. Unlike Flex, you cannot simply use a control's property as a data binding source in the markup. [6]

3.5 UI Components

The UI components of each framework seem to match quite nicely with each other, thus it is not necessary for me to go into the details. Much of this UI component mapping is completed already. [7]

4 Academic Work

4.1 Portability

One of the key aspects in Rich Internet Applications is portability, the ability for these RIA applications to be working in different systems and machines. It is surprising that portability is even an issue in this modern ear, as one would have thought the ability to run Internet browser-based applications on cross-platforms should be already in place. I think this is largely because commercial issues, rather than being technically related.

Flex/Flash

Flash being the most well-known and generalized RIA framework out on the market. It is normal to think it is widely supported, and a Flex or Flash application should be able to run on almost any platforms or browsers.

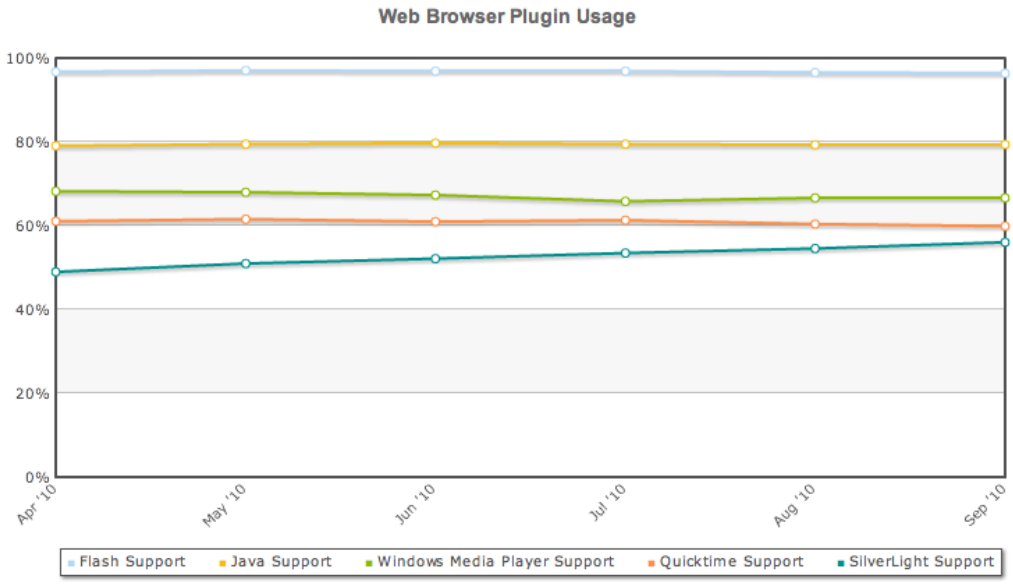


Figure 6 Web Browser plugin market share [8]

Flash is supported in almost all operating systems, including Windows, Mac OS 9/X, Linux, Solaris, HP-UX, Pocket PC/Windows CE, OS/2, QNX, Symbian, Palm OS, BeOS, and IRIX.[9] There is also an lightweight version of Flash called 'Adobe Flash Lite' to fit the mobile environment restrictions, and allows users of mobile devices to view multimedia content and applications developed using Adobe's Flash tools, which had previously been available only on personal computers. However, due to possible commercial issue, Apple does not allow third-party runtimes on its mobile device products (iPhone, iPad), which accounts for more than 60% of global smartphone web traffic, or the iPod touch, which makes up more than 95% of "mobile Internet device" traffic. This limits Adobe's ability to market Flash as a ubiquitous mobile platform. However, Flash support has been announced for competing mobile platforms, including the next version of Android. [10]

Silverlight

Silverlight being the latecomer in the RIA industry has rather limited support for desktop and mobile platforms. Silverlight is supported fully in Windows, Mac OS, under IE 6,7,8 Firefox 3, Safari (via NPAPI), Google Chrome, and supported on mobile devices, Windows phone 7 devices. However, on the latest information coming from Microsoft, they "don't have the extensibility model inside the browser for those pieces [Silverlight, Flash, HTML5]" [11]

Unfortunately, Silverlight is not officially developed and supported in the Unix-based environment, maybe partially because the market in the UNIX community is not large enough for Silverlight to commit the development work yet. However, Silverlight is available on Linux & UNIX-based OS (SUSE Linux Enterprise Desktop 11, openSUSE 11.x, Ubuntu 9.10, and Fedora 12) through 'Moonlight'. [12]

Moonlight

Moonlight is an open source implementation of Microsoft Silverlight, mostly done by a group of people in Novell[13]. The existence of such project is to provide the support of Silverlight on the Linux and other Unix/X11 based operating systems (in the form of Firefox and Chrome browser plug-in)[14], which Microsoft was not aiming to be a platform for their Silverlight applications. However, shortly after the moonlight had its public release, Microsoft agreed to

work with the Novell Team to support the Moonlight development. Although Silverlight is now supported on Unix machines through Moonlight, it is developed through 'tracking' and implementation based on the release of a new Silverlight version, thus it's always one step slower (Silverlight 1.0 released on September 2007, Moonlight 1.0 released on Jan 2009; Silverlight 2.0 released on October 2008, Moonlight 2.0 released on December 2009; Silverlight 3.0 released on July 2009, Moonlight 3 planning to release by the end of 2010)

4.2 Performance

Performance plays an important part almost in everything we do in the computation world. The world is constantly looking into ways that we can do things faster, run a task quicker. In this section, we look into the performance of the two RIA frameworks, Silverlight and Flex, using some benchmark programs in the RIA community and the current trend in the computing world to improve performance—multi-threading/multi-processing.

4.2.1 BubbleMark

What is BubbleMark?

BubbleMark is a specific animation program that will test and compare the performance of different RIA frameworks in different Internet browsers. This program is implemented in varies frameworks, including JavaScript/Dynamic HTML, Silverlight, Flash/Flex, and Java. For benchmark testing, the equivalent program is run using different RIA frameworks and record the frame-per-second result of the 2D animation, which simply are numerous balls moving around rapidly.

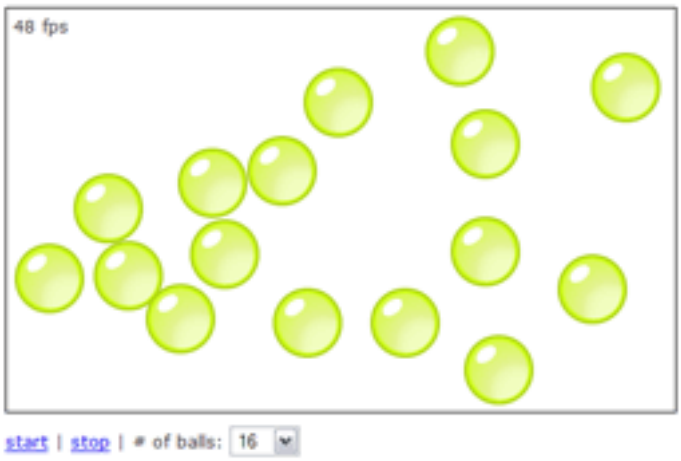


Figure 7 A screenshot of a running BubbleMark program

Why not BubbleMark?

There are numerous discussions among the RIA community about why BubbleMark is not a good benchmarking tool for measuring RIA framework performance. Below, I will give a few of those points.

- **Incorrect implementation of the BubbleMark implementation in Flash version.** This is caused by the misconception and misuse of the 'Flash' rendering engine. The Flash BubbleMark program generates incorrect animation rendering results, as "Flash holds on to all display updates till the next render pass and applies all the latest changes at once". [15] This means having object movement rate much quicker than the animation update done by the rendering engine is completely pointless as it will not be reflected on the frame-per-second rendering result.

- **Having test applications that will generate frame rates higher than 60 fps are meaningless.** This is because the majority of computer users these days are on LCD monitors, which natively run at 60 fps. Also, on certain operating system like, Mac, will limit the effective frame rate of paint requests it sends to the graphic card.[16] Thus, when results from BubbleMark happens to be much higher than 60 fps, it is likely that the extra frames are never even calculated by rendering engine of the RIA framework. In reality, we will only be seeing a maximum of 60 frames per second anyway.

- **The nature of the program has little meaning on the big picture.** This benchmark program only tests one simple aspect of the rendering engine in

these technologies, which is bitmap translation. At the most, what we will be able to obtain is the performance of how each framework is able to achieve running bitmap particle emitters. Also, it doesn't take into account the processor power that is used to run these programs, some testing were done, and have observed a rather low frame-per-second, yet the CPU usage is relatively low as well. This means that the CPU is taken full advantage of, in generating these results—using less power over faster motion.

4.2.2 GUIMark

What is GUIMark?

GUIMark was produced largely because people easy the weaknesses and flaws in the BubbleMark animation program created before. GUIMark is a benchmark test suite designed to compare the rendering systems of RIA frameworks. In general it should be able to give designers and developers a good indication of which technologies can draw complex interfaces at a smooth rate of motion [17]. The test mostly addresses RIA technologies like Flash, Silverlight, HTML or Java, but was designed to be easily ported to any 2D GUI environment. This program is based the BubbleMark, but was designed to heavily saturate the rendering pipeline and determine what kind of visual complexity is achievable in the sub-60 fps realm.

Important facts and assumptions about the benchmark-testing program [18]:

- The max draw rate for an application is fixed at 60 frames per second.
- Animations should run on fixed time as opposed to relative time.
- The test case should be hard to optimize.
- Code execution should have a negligible impact.
- The rendering pipeline should remain saturated.

GUIMark results (2008)

From the result done by the RIA community back in year 2008 using previous versions of the RIA frameworks—Flex 3 and Silverlight 2, we can see that Flex out-performances Silverlight by a large margin. (Especially in windows environment) The performances of these RIA applications are generally better in the Windows operating system environment, may be due to the hardware acceleration that windows provides. [19]

Testing environment:
OS: Windows XP, Mac OS X 10
CPU: Intel Core 2 Duo 2.33 GHz
RIA framework: Flex 3, Silverlight 2

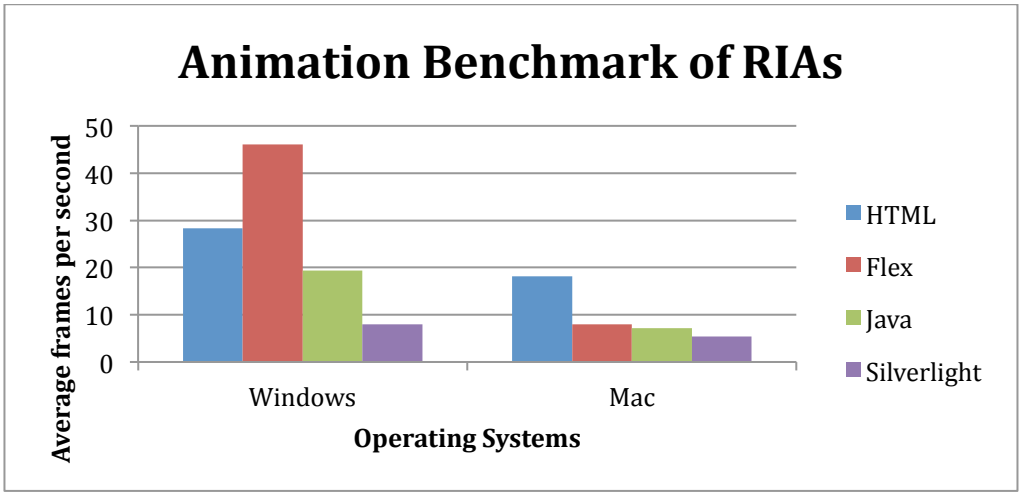


Figure 8 Graph result of GUIMark (2008)

GUIMark Results (2010)

Since the result obtained was tested using previous version of Silverlight and Flex/Flash framework, thus I thought it will be meaningful to do the comparison between the newest version of the RIA frameworks, since the rapidly growth in the RIA technology in recent years, especially Silverlight.

I have converted the GUIMark testing applications into the newest Flex and Silverlight framework versions, and conducted the benchmarking using my MacBook Pro computer.

From the graph result below, for this particular test we can see that the newest version Silverlight (version 4) has significantly improved its animation performance since the version 2 release two years ago. In the new results, the average animation performance of Flex and Silverlight in the major web browser sin the Windows 7 environment is almost identical to each other. However, in the MAC OS X environment Silverlight actually out-performs Flex.

Testing environment:
OS: Windows 7, Mac OS X 10.6
CPU: Intel Core 2 Duo 2.66 GHz
RIA framework: Flex 4, Silverlight 4

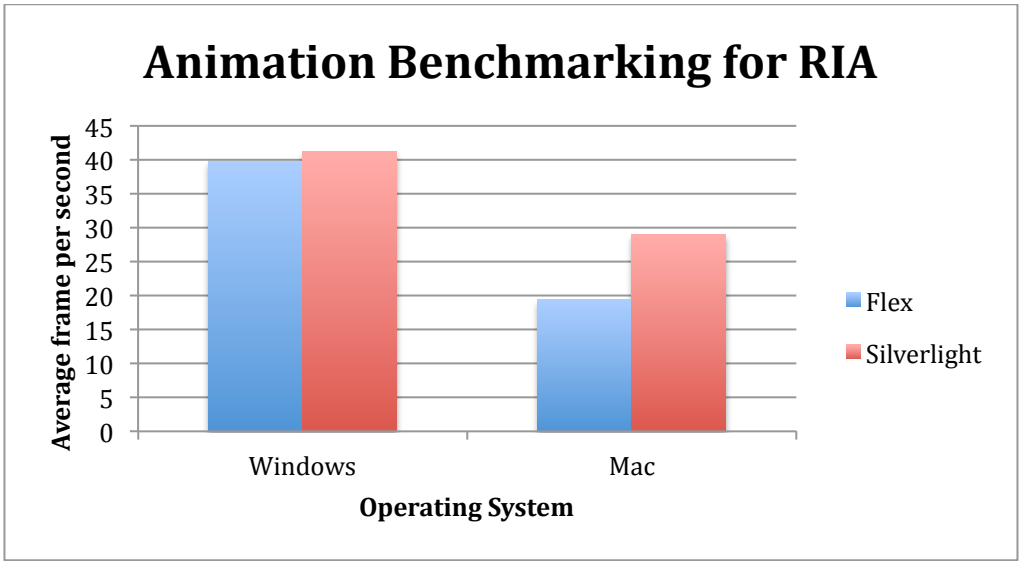


Figure 9 Graph result of GUIMark (2010)

Limitation

Microsoft Internet Explorer not included

Microsoft Internet Explorer was left out in the benchmarking test conducted by me, simply because it is not cross-platform. Since, the test conducted was a measure between the two Operating systems; it would be unfair for the windows results to include the performance.

However, even with the forever-decreasing Microsoft Internet Explorer market share, it still has a very large customer group, around 50% globally [20]. This could largely impact the average frame-per-sec metric tested across the different major Internet browsers. Especially taking into account that the same company—Microsoft, develops both Internet Explorer and Silverlight.

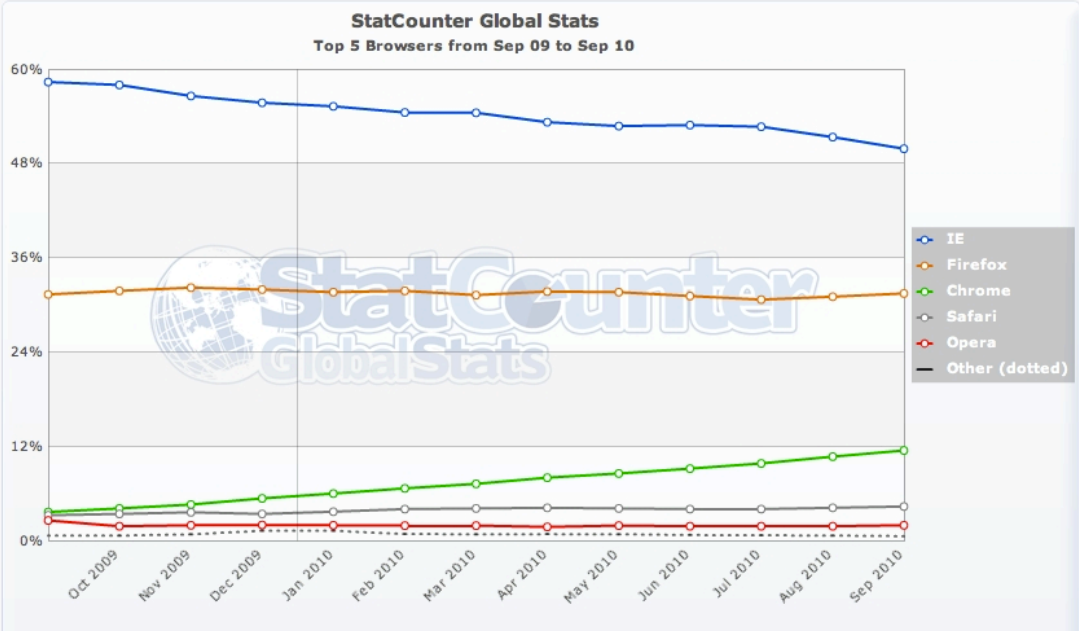


Figure 10 Global Internet Browser market share stats [21]

4.2.3 Multi-Threading

What is multi-threading?

Multi-threading in simple terms is basically trying to do more than one task at a time within a process. Much like processes are running parallel in an operating system, playing music files and doing word processing at the 'same time', the same could be applied in threading within a single process.

In Silverlight development, Microsoft has really taken full advantage of its enormous .NET framework. The way that multi-threading is implemented in Silverlight applications is complete the same way as if you are to implement multi-threading in any ordinary C#, WCF, WPF applications.

Silverlight the winner?

This is really one major area that Silverlight has a big step ahead of Flex, and making up some of the lost grounds in other aspects of the frameworks. There are multiple ways of implementing multi-threading in .NET applications, such as using a thread pool, and background worker. For this particular testing I have

conducted, I chose to use background worker. In the program created, each thread is represented by a background worker class object.

Flex

Flex framework is a little behind Silverlight when it comes to multi-thread. In another words, Flex framework does not multi-thread. However, it is still to create thread-like behavior programs, 'pseudo-threading' [22], but you will never be able to achieve the multi-thread performance. The application developed using this pseudo-threading, will only 'act' like a multi-threaded application, such as the user interface will not be frozen when the application is computing some heavy loaded task in the background. (Fooling the user to think that the user interface and the computation are each handled separately by multiple threads)

Basically, the way this work is to divide a heavy computation process or task into several smaller portions. However, this would have to be done manually by the developers themselves. This can be difficult and complex depending on the nature of the program itself, as sometimes tasks can be easily divided up or the efforts spend in dividing these tasks are simply not worth the time and effort spend.

The important point in construct such an feature is to ensure that the portions of the computation is divided into small enough parts that the user interface response time and restoration of the execution context can be done effective. Otherwise, the program will spend the majority of time moving in and out of the 'threads' instead doing the intended tasks.

Alex Harui, a Flex specialist, has implemented an example of such pseudo thread. The concept behind his implementation is that a PseudoThread instance takes a callback function and a context object and calls the callback function with the context object as many times as it thinks it can get away with it within the frame interval. The callback function should update the context object before returning and return false when done and the PseudoThread instance dispatches an event and destroys itself.

Test conducted

I have created a program that computes nested loops calculation, and the same equivalent program is implemented in both Silverlight and Flex frameworks. The program itself is constructed in the way that parallel computing can be achieved,

thus no dependencies within the nested loops (the main calculation) that would stop the threads from running parallels.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    int n = Int32.Parse(this.inputTextBox.Text);
    DateTime start = DateTime.Now;
    int i;
    int j;
    int numprimes;
    double limit;
    numprimes = 1; //2 is prime

    for (i = 3; i <= n; i += 2)
    {
        bool isPrime = true;
        limit = Math.Ceiling(Math.Sqrt(i)) + 1;
        for (j = 3; j < limit; j += 2)
        {
            if (i % j == 0)
            {
                isPrime = false;
                break;
            }
            if (isPrime != true)
            {
                continue;
            }
        }
        if (isPrime)
        {
            numprimes++;
        }
    }

    DateTime end = DateTime.Now;
    Double timetaken = end.Ticks - start.Ticks;
    timetaken = timetaken / 10000000;
```

Figure 11 For Loop computation (Silverlight)

4.2.3.1 Implementation

Flex application

The implementation of the flex application is straightforward, using MXML to define the UI controls, and implementation the for loop computation in the ActionScript file. Thus, when the calculation button is pressed, the application UI will be frozen as the code in behind is trying to compute the result of the for loop code. Once the computation is finished, the time taken will be displayed on the application user interface.

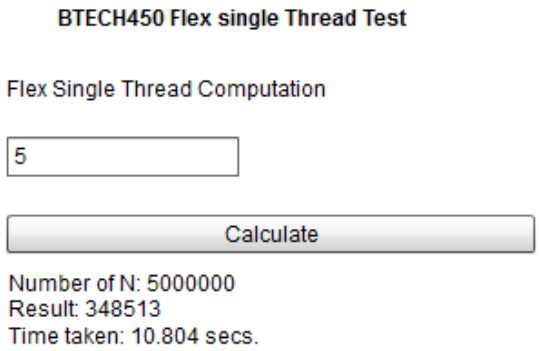


Figure 12 Flex single thread application

Silverlight Application

The implementation of the Silverlight application is much complicated compared with the flex application (single thread). Basically, depending on the number of threads that we are testing (in this particular application, the number of threads are fixed to one, two, or three threads), the same number of background work object will be initialized, and called to execute one proportion of the entire for loop calculation. Similar to the Flex application, the time taken to complete the computation will be returned to the application UI. However, the only difference is that when the application is running on two threads or three threads, the UI will not be frozen during the time of computation. This is because the UI and the computation are running on different threads.

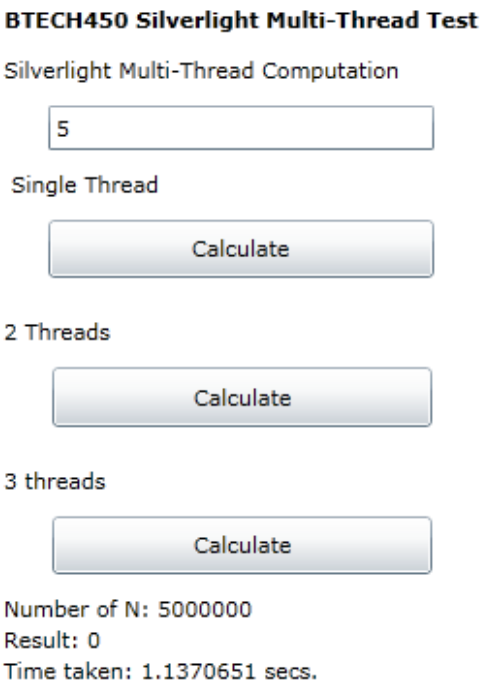


Figure 13 Silverlight Multithread application

Limitation

Like what is stated before, this test is largely depended on the nature of the program—low dependencies between processes. A program with high dependencies within the codes will decrease the performance of the multi-threading implementation, as more proportion of the code will need to be executed sequentially (single thread execution).

Another limitation of the test conducted is that this testing is merely testing the performance of the RIA frameworks in heavy load computations in for loops. This doesn't necessary mean that the same result will be generated for other type of computation, .e.g. animation, and video playback.

Result

From the graph shown below, we can see that in this particular test, even the singled threaded Silverlight by far, out performances the single-threaded Flex application. Also, we can see that the Silverlight application, running on two threads or three threads, each have significant decrease in time taken to complete the computation.

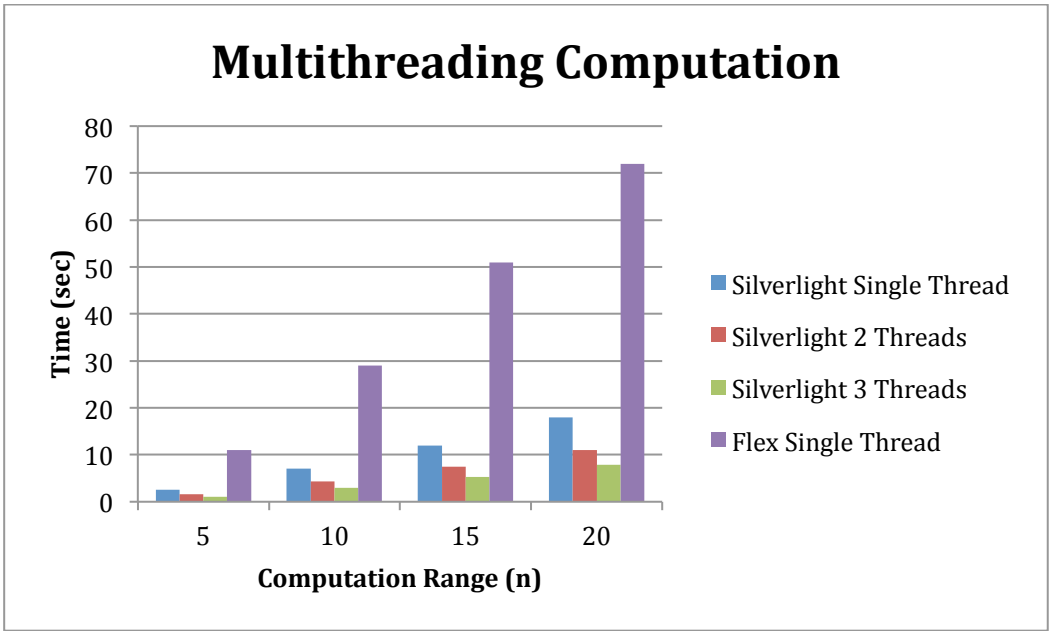


Figure 14 Graph of Silverlight and Flex applications computation time

4.3 Security

The focus of the RIA security research for this project was on how different frameworks confirm that the clients/users are who they claim to be (authentication), and confirming if an authenticated user has the permission to perform a particular task or have access to a particular resource (authorization). In addition, we looked into some security issues that are specific to RIA applications.

Regarding to security, the project looked into the two common security approaches in software—declarative and programmatic. Declarative security often refers to the configurations and mechanisms applied on the server side to ensure the resources on the server are protected.

Due to the nature of RIA applications, the source code for the application is no longer run on the server side (JavaScript & ASP.NET codes are executed on the server). Most of the time the client has to download the Silverlight or Flex application, and run it on the client machine.

This exposure of application source code, puts the security of the application in danger, and greatly increases the chance of hackers modifying the code, and knowing the underlying structure of the application.

Using reflector products like '.NET Reflector', this will allow the compiled Silverlight assemblies to be converted back to C# source codes, and enabling hackers and programmers to 'view your code, variable names, strings values, connection strings and username/password combinations' [23]. Thus, allowing them to bypass the security parameters of the server using the modified version of the application.

*For Flex platforms, there are also similar technologies, such as Flash de-compiler, which will transform SWF files back to the FLA source code files. Cause of the existence of such technology, it is extremely important not to 'store connection strings or username/password combinations' inside the RIA codes, do not 'directly connect to your domain database' from your application code, 'always use a web service'. By does this, even if the hacker manages to obtain the source code, it is not possible for them to launch attacks, such as DDoS attacks, to the server or database

For a Silverlight environment instead of trying to make the Silverlight application code itself secure (impossible), the effort should be used to protect the web servers that are hosting the 'services' that the Silverlight application interacts with.

*There are already free & commercial software products available on the market that can de-compile the compiled Silverlight code files (.xap), such as Silverlight Spy.

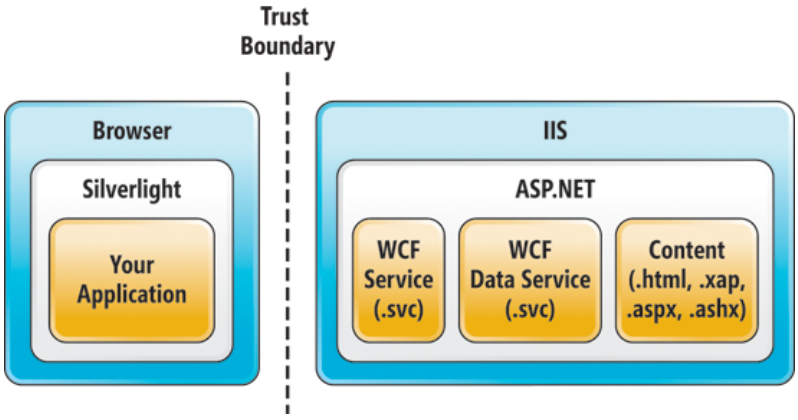


Figure 15 Silverlight Architecture

4.3.1 Authentication

Silverlight

Authentication in Silverlight uses the same methodologies as most other .NET applications. The two most common authentication methods in the Microsoft platform are Windows authentication and forms authentication.

Windows authentication

Windows authentication leverages the Local Security Authority or Active Directory to validate user credentials. This is a big advantage in many scenarios; it means you can centrally manage users with tools already familiar to systems administrators. Windows authentication can use any scheme supported by IIS including basic, digest, integrated authentication (NTLM/Kerberos) and certificates.

The integrated scheme is the most common choice for use with Windows authentication, because users don't have to provide their user names and passwords a second time. Once a user logs on to Windows, the browser can forward credentials in the form of a token or a handshake that confirms the person's identity. However there are also disadvantages to using integrated authentication, because both the client and server need visibility of the user's domain. As a result, it's best targeted at local networks. Furthermore, though it works with Microsoft Internet Explorer automatically, other browsers, such as Mozilla Firefox, require additional configuration.

Both basic and digest authentication typically require users to re-enter their user names and passwords when they initiate a session with your Web site. But because both are part of the HTTP specification, they work in most browsers and even when accessed from outside your organization.

Silverlight leverages the browser for communication, so Windows authentication is easy to implement with any of the IIS authentication methods just discussed.

Form authentication

Forms authentication is a mechanism that provides simple support for custom authentication in ASP.NET. As such, it's specific to HTTP, which means it's also easy to use in Silverlight.

The user enters a user name and password combination, which is submitted to the server for verification. The server checks the credentials against the database, and if they're correct, returns a FormsAuthentication cookie. The client then presents this cookie with subsequent requests. The cookie is signed and encrypted, so only the server can decrypt it—a malicious user can neither decrypt nor tamper with it.

Exactly how you invoke forms authentication varies depending on how you implement your login screen. For example, if you've used an ASP.NET Web form that redirects to your Silverlight application after the user's credentials have been validated, you probably have no more authentication work to do. The cookie already will have been sent to the browser and your Silverlight application will continue to use the cookie whenever making a request to that domain.

If, however, you want to implement the login screen inside your Silverlight application, you'll need to create a service that exposes your authentication methods and sends the appropriate cookie. Fortunately, ASP.NET already provides what you need—the authentication service. You just need to enable it in your application.

Another great feature of ASP.NET authentication is its extensibility [24]. A membership provider describes the mechanism by which the user name and password are verified. Fortunately, there are a number of membership providers available as part of ASP.NET, including one that can use SQL Server databases and another that uses Active Directory. However, if a provider that meets your requirement isn't available, it's straightforward to create a custom implementation.

Flex

Authentication in a server environment under Flex is achieved through the J2EE implementation [25]—Java Authentication and Authorization Service (JAAS), Java security manager and policy files. The authenticating mechanism applied is role-based, means that all users wanting access to a web application is represented by one or more roles. Before a user can be granted access to a web application resource, the container ensures that the user is identified (according to his/her login details), and that the user has the appropriate access level to obtain the requested resource. Any unauthorized access of a web application results in an HTTP 401 (Unauthorized) status code. Authentication requires a website to store information about users.

Usually, a websites that authenticate user access typically implement a login mechanism that forces verification of each user's identity by using a password. After the website validates the user, the website can then determine the user's roles.

Authentication occurs on a per-request basis. The container typically checks every request to a web application and authenticates it. Authentication requires that the roles that the application developer defines for a web application be enforced by the server that hosts the application. The web application's deployment descriptor, web.xml, contains the settings for controlling application authentication. This file is stored in the web application's WEB-INF directory.

Constraints can be set up to prevent unauthorized access to your application. There is a choice in between protecting the page that the Flex application is returned with, or protecting the SWF (application) file itself. This can be achieved by defining specific URL patterns in the web.xml file. For example,

```
<web-app>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>BTECH Application</web-resource-name>
                <url-pattern>/websiteAccess/*</url-pattern>
                <http-method>GET</http-method>
                <http-method>POST</http-method>
            </web-resource-collection>
        <auth-constraint>
            <role-name>supervisor</role-name>
        </auth-constraint>
    </security-constraint>
</web-app>
```

When the browser tries to load a resource that is secured by constraints in the web.xml file, the browser will require the user to enter username and password if BASIC authentication is implemented for the application or forwards the user to a login page if FORM authentication is implemented.

BASIC authentication Configuration

BASIC authentication can be configured by using the login-config element and its auth-method subelement in the web application's web.xml file, as the following example shows:

```
<web-app>
    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>Supervisor</realm-name>
    </login-config>
    ...
</web-app>
```

FORM authentication configuration

With FORM authentication, a webpage must be implemented to request username and password from the user, and submit them as FORM variables. This form can be implemented in HTML or as a Flex application or anything that can submit a form. For example,

WEB.XML FILE

```
<web-app>
    <login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
            <form-login-page>/login.htm</form-login-page>
        </form-login-config>
    </login-config>
</web-app>
```

HTML FORM

```
<form method="POST" action="security_check">
    <table>
        <tr><td>User</td><td><input type=text
name="username"></tr>
        <tr><td>Password</td><td><input type=password
name="password"></tr>
    </table>
    <input type=submit>
</form>
```

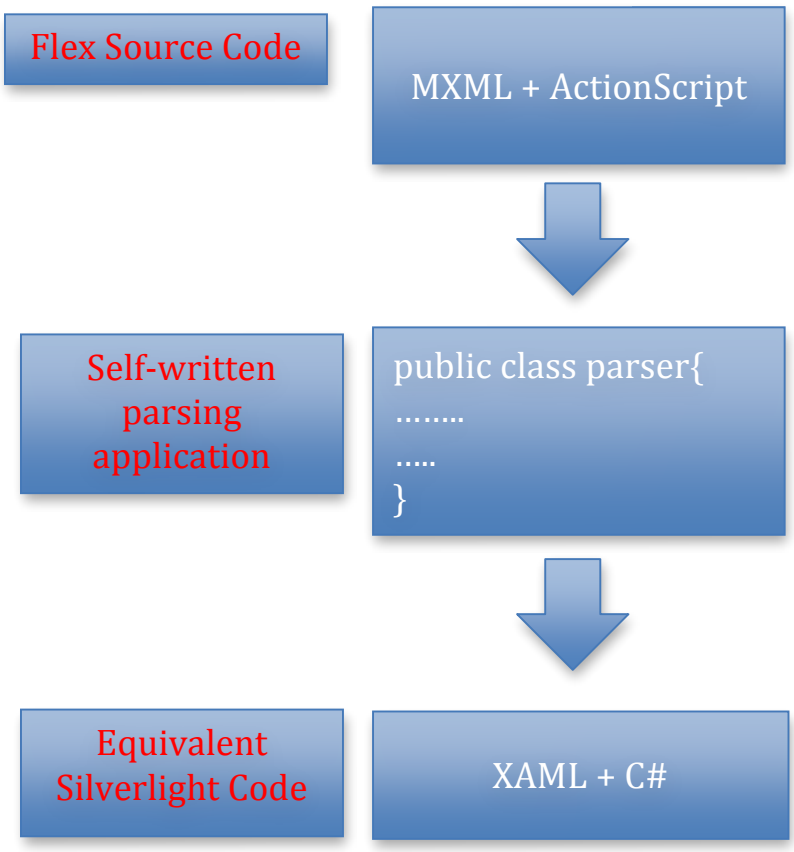
The result from the HTML form is submitted to the container's JAAS system with base-64 encoding, [25] which means they can be read by anyone that can view the TCP/IP traffic. In order to prevent attacks eavesdropping on the packets send over the network, encryption is necessary to prevent these so-called "man-in-the-middle" attacks. In both BASIC and FORM authentication, if the user accessed the resource through SSL, the username and password submission are encrypted, as is all traffic during that exchange. After server receives the request, the container populates the browser's security context and provides or denies access to the resource. Flash Player inherits the security context of the underlying browser. As a result, when you make a data service call, the established credentials are used.

5 Auto Code Generator

As one of the ultimate goal of this project, the aim was to build an auto code generator that will take a flex application code as the input, and output an equivalent Silverlight version of the application.

The original idea was to build a parser application using either C# or Java completely by myself which will break down a Flex source code, and translate it into Silverlight code line by line.

Original Concept Diagram



However, after looking into the details of each language, and the information gathered from previous phases of the project, I soon realized that this was not

possible to do. Cause, although the two RIA languages have a lot in common in terms of what functionalities they provide, and the structure of files used to build the applications (Markup file for interface and Code files for application & business logic).

In order to do this code generator correctly (i.e. maintaining the structure and style of original Flex code), the academic mentor Xin Feng Ye, suggested to use a compiler generator to do this. Compared to a hand-built recursive-descent parser, table-driven/parsers often do not have enough parsing strength and can be difficult to understand and debug.

A parser must do much more than just recognize languages. In particular, parsers must interact with the lexer, report parsing errors, construct abstract syntax trees, and call user actions. Existing parsing tools have focused mainly on the language recognition strategy, often ignoring the aforementioned tasks. [26]

Xin Feng had suggested me to use the Coco/R as it was the compiler tool used by one of his supervising master students, but I went and researched on the other possible compiler tools. In the end, I found ANLTR compiler tool. I chose ANTLR over Coco/R solely because I was able to find more resources and materials on ANTLR than Coco/R, not because technical reasons. This was quite important as no matter which one I chose, I had to learn the tool by myself from scratch.

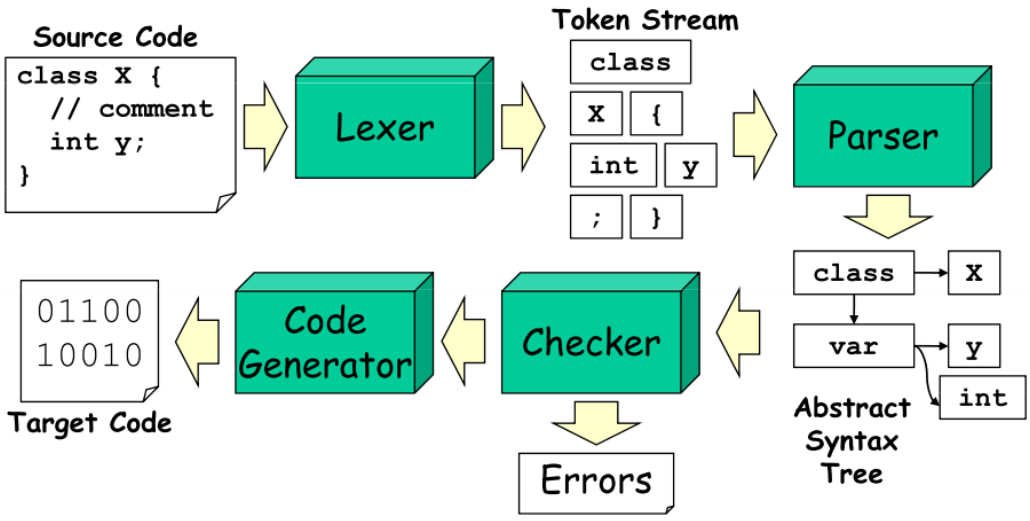


Figure 16 Compiler Tool Workflow Diagram

5.1 Coco/R

Coco/R is a compiler generator, which takes an attributed grammar of a source language and generates a scanner and a parser for this language. The scanner works as a deterministic finite automaton. The parser uses recursive descent. LL (1) conflicts can be resolved by a multi-symbol look ahead or by semantic checks. There are versions of Coco/R for different languages, including C#, Java and C++. [27]

5.2 ANTLR

Introduction

ANTLR stands for Another Tool for Language Recognition. It is a parser generator based on LL (*) parsing. It operates by taking in a user-defined grammar that specifies a language, and generates the source code for a recognizer for another language. Currently, the ANTLR supports generated codes in several mainstream programming languages, such as C language, Java, C#, Objective-C, and Python. This means that the parser/language recognizer it is able to generate can be in any of those four languages.

By default ANTLR reads a grammar and generates a recognizer for the language defined by the grammar, (i.e. a program that reads an input stream and generates an error if the input stream does not conform to the syntax specified by the grammar). Actions can be attached to grammar rules in the grammar. These actions are written in the programming language that the recognizer is being generated in (for this project, the 'actions' are the statements to generate the Silverlight equivalent codes). When the recognizer is being generated the actions are embedded in the source code of the recognizer at the appropriate points.

From the screenshot of the ANTLR tool (below) running in the Eclipse IDE software, we can see a few java application files (i.e. BTECH.java), grammar files (i.e. BTECH.g), and ANTLR generated lexer and parser (i.e. BTECHLexer.java and BTECHParser.java)

The java application file, BTECH.java, is basically the main auto generator that reads in the input source code from a Flex file, and using the ANTLR generated lexer and parser, it is able to produce the corresponding Silverlight code.

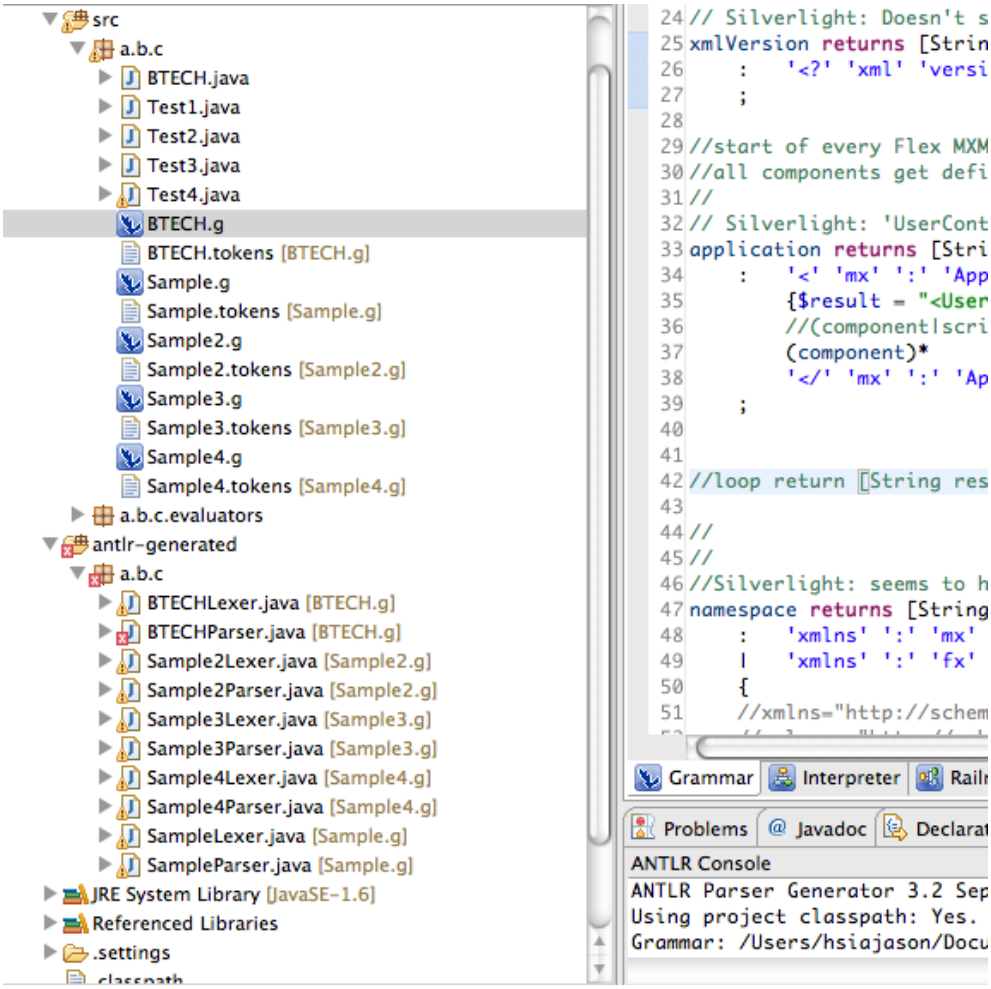


Figure 17 Screenshot of ANTLR under Eclipse IDE

Lexer

Lexer is one part of the two major components that ANTLR generates. It is also known as a 'scanner'. Basically, for all lexers, it needs to break the input text into parser recognizable token objects.

Basically, for this project, the input source will be Adobe Flex code files. These source files will be streamed into the lexer character by character, and the job of the lexer is to break the meaningless stream of characters into separated individual pieces. Each of these pieces is a token, and is separated out according to the grammar defined by the user. These tokens are components of the input source programming language, such as keywords (e.g. 'for', 'while',

'break'), variable identifiers (e.g., 'a', 'b', 'c'), symbols and operators. (e.g. 'll', '&&').

The lexer will ignore comments and unrelated characters such as whitespace from the input source code, and then the lexer will convert the stream of characters into a stream of token [26] which have individual meanings as dictated by the lexer grammar rules. This stream of tokens generated by the lexer is received by the parser.

When the lexer can't identify the sequences of characters to the token types defined by one of the rules in the grammar file (in ANTLR this is user-defined), then it will generate errors.

Parser

A parser is one of the components in a compiler, which checks for correct syntax and builds a data structure (abstract syntax tree) implicit in the input tokens. The parser often uses a separate lexer (as explained previously) to create tokens from the sequence of input characters.

The type of parser that ANTLR generates is *LL parser. This basically means that ANTLR generates a parser that performs top-down parsing for context-free grammars. It parses the input source code from left to right.

The parser uses the languages are described by a grammar (user-defined), the grammar determines exactly what defines a particular token and what sequences of tokens are decreed as valid. The parser organizes the tokens it receives into the allowed sequences defined by the grammar of the language. If the language is being used exactly as is defined in the grammar, the parser will be able to recognise the patterns that make up certain structures and group these together.

Grammar

The main focus of my project is the produce a Flex language grammar that is based on my knowledge of the Flex programming language. This will act as the grammar of my instance of ANTLR tool. I have manually converted the basic programming syntax, data structure, programming style of the Flex language into separate grammar rules and expressions.

In this project, I have defined the some basic grammar structure for Adobe Flex, mainly in the areas of the MXML markup UI content, and simple ActionScript structures like for-loop, if-statements. It does not contain complex

structure like framework library mapping, cause these are extremely hard to convert correctly.

The grammar file itself contains all the rules that match the Flex syntax, and each of them includes some inline code that will specific the Silverlight output code for each matching Flex syntax rule. This grammar rules structure will allow me to handle recursion and nested relationships within components very well. In addition, it will keep the parsing structure tight and robust. Whereas, compared with a self-constructed parser proposed in the beginning, the data structures will be impossible to handle and keep track of when the code structure becomes larger and more complex.

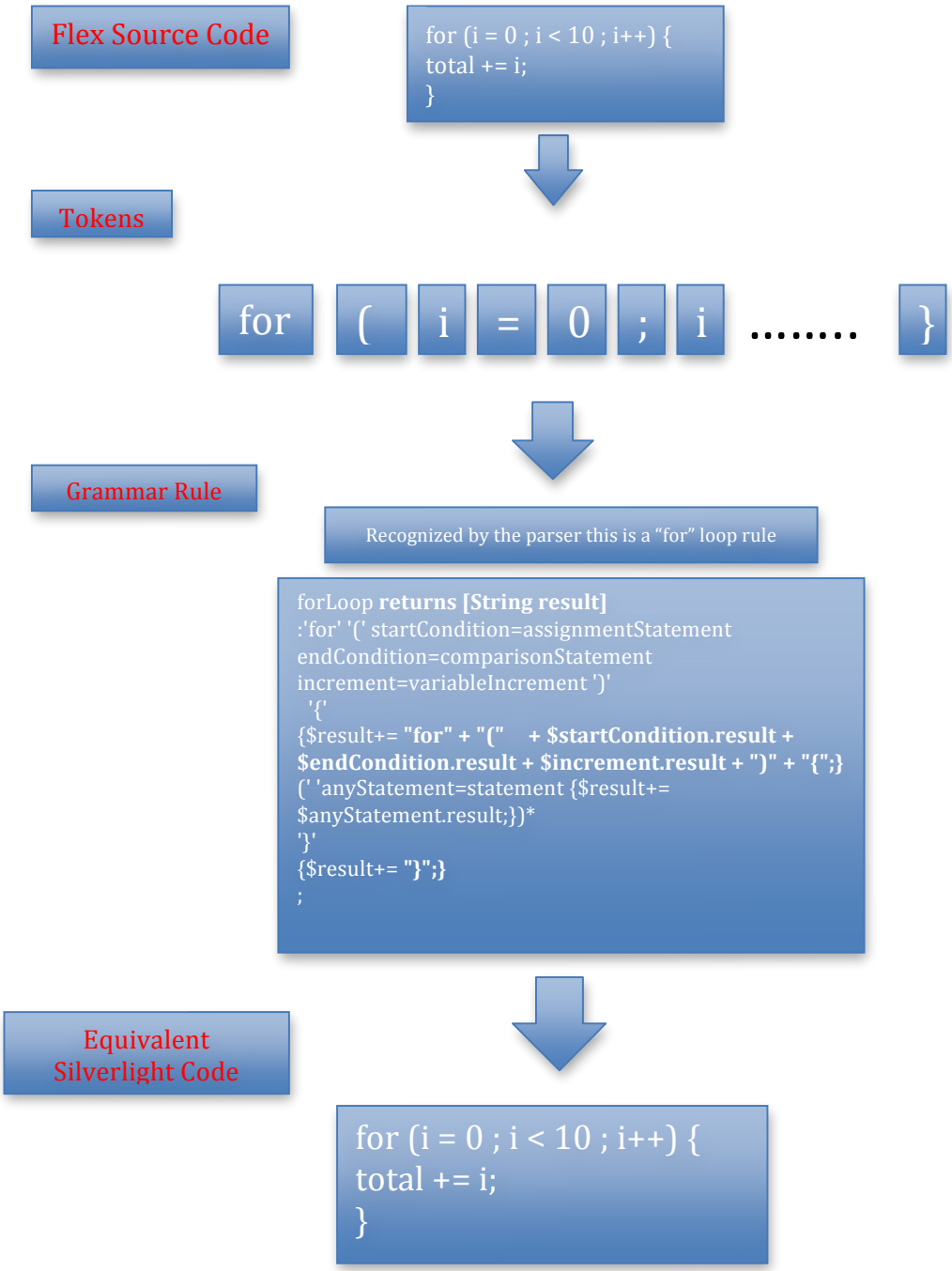
Below are the steps and an example of how a 'for' loop rule is applied in an ANTLR tool runtime execution.

How does it work?

Below is an example workflow on the whole process of converting a 'for' loop flex source code into Silverlight code using the ANLTR compiler.

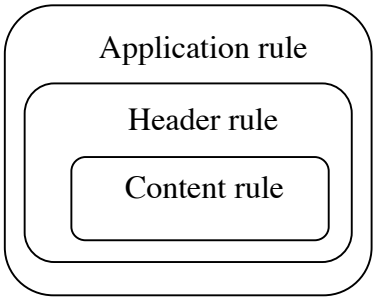
STEPS:

1. Flex source code is divided into token stream
2. The tokens in the stream are being matched will the rules defined in the grammar.
3. Once a matching grammar rule is identified, it generates the output (Silverlight code) according to the code defined in the grammar rule.


Flex Source Code
for (i = 0 ; i < 10 ; i++) {
total += i;
}
Tokens
for
(
i
=
0
;
i
.....
}
Grammar Rule
Recognized by the parser this is a "for" loop rule
forLoop returns [String result]
:'for' '(' startCondition=assignmentStatement
endCondition=comparisonStatement
increment=variableIncrement ')'
'{'
{$result+= "for" + "(" + $startCondition.result +
$endCondition.result + $increment.result + ")" + "{";}
(' 'anyStatement=statement {$result+=
$anyStatement.result;})*
'}'
{$result+= "}";}
;
Equivalent
Silverlight Code
for (i = 0 ; i < 10 ; i++) {
total += i;
}

Challenges

One of the major issues I had is mapping the different programming syntax of the same data structures between the two RIA languages. When this occurs, it is extremely hard to deal with, because the way ANTLR works, is that the output code (Silverlight) is defined within each grammar rule that maps the Flex syntax. It is impossible to pass on additional information between rules other than the result code of one rule to its encapsulating rule (shown below).



The solution proposed by Xin Feng, my academic mentor, was to save all the mismatching information between two framework syntax in a separate data structure. This only worked to a point where the information can be abstracted saved, but it was impossible for me to place all these information into the location of the output file containing the Silverlight code.

6 Conclusion

From the research and development work I have done on the two RIA frameworks—Silverlight and Flex. I managed to put out a simple guideline that looks into the main areas of each framework and how to construct programs that will make the transition from Flex to Silverlight much easier. Ultimately, I was able to develop an auto code generator prototype that allows users to convert Flex code to Silverlight code.

As a side note, Silverlight has really came a long way to match Flex's framework features, such as the out-of-browser application feature, and in some areas have already overtaken Flex—multithreading. However, the discussion of which is the better framework remains, as Silverlight has major disadvantages in it portability, being Windows-only development environment, and relying on their parties to support Silverlight on Unix-based machines.

6.1 Future work

Due to the complexity of the auto code generation from one framework to framework, major work remains to be done, as there is no product out on the market that will fully convert a complex Flex application into an equivalent Silverlight application. Partially, due to the differences in their own code libraries and framework structure. Also, as the result of this it will be interesting as how an error detection function can be implemented in conjunction with the code generator to help to identify the differences between the original Flex source code, and the generated Silverlight code.

Another interesting point not covered extensively is the development of HTML5, which some view it as the RIA killer. However, not much in-depth comparison of this technology is research. Thus, it will be really interesting to look into the different ways of how well HTML5 competes with the major RIA frameworks in the areas that are discussed in the report, such as security, performance, and usability.

7 References

1. Moses, J.J “Where the Wild Things Are: an Adventure in Silverlight (Act I)”, http://insideria.com/2010/03/where-the-wild-things-are-an-a.html, InsideRIA , 2010
2. "Web Browser Plugin Market Share". StatOwl. http://statowl.com/plugin_overview.php. Retrieved 2009-08-18.
3. Schmidt W., “XAML Overview”, http://msdn.microsoft.com/en-us/library/cc189036(VS.95).aspx , April 2010
4. Moses, J.J, “Where the Wild Things Are: an Adventure in Silverlight (Act III)”. InsideRIA, http://insideria.com/2010/04/where-the-wild-things-are-an-a-2.html, 2010
5. Ezell J., “Silverlight vs. Flash: The Developer Story”, http://weblogs.asp.net/jezell/archive/2007/05/03/silverlight-vs-flash-the-developer-story.aspx, 2007
6. Moses, J.J, “Where the Wild Things Are: an Adventure in Silverlight (Act IV)”. InsideRIA, http://insideria.com/2010/04/where-the-wild-things-are-an-a-3.html, 2010
7. Project Rosetta, “Flex Components to Silverlight Controls: API Guide”, http://visitmix.com/labs/rosetta/FTSL/Guide/FlexComponents/, 2009
8. Stat owl, “Web Browser Plugin Market Share/ Global Usage”, http://statowl.com/plugin_overview.php, 2010
9. Wikipedia, “Adobe Flash”, http://en.wikipedia.org/wiki/Adobe_Flash
10. Nickinson, P., “Andy Rubin says Flash is coming in Froyo version of Android operating system”, http://www.androidcentral.com/andy-rubin-says-flash-coming-froyo-version-android-operating-system, androidcentral, April 2010
11. Barribeau, T., “Why No Flash Or Silverlight on Windows Phone 7?”, http://www.everythingwm.com/why-no-flash-or-silverlight-on-windows-phone-7/2010/10/15/, October 2010
12. Wikipedia, “Microsoft Silverlight”, http://en.wikipedia.org/wiki/Microsoft_Silverlight, 2010
13. Icaza, M., Anguiano, J., Sanchez, L., “FAQ: General”, http://mono-project.com/FAQ:_General
14. Mono, “Moonlight Supported Platforms”, http://mono-project.com/MoonlightSupportedPlatforms
15. Chirstmann S., “Why Bubblemark is a poor ui benchmark”, http://www.craftymind.com/2008/04/11/why-bubblemark-is-a-poor-ui-benchmark/, April 2008
16. Beijnum I., “Beam sync: friend or foe”, http://arstechnica.com/apple/news/2007/04/beam-synchronization-friend-or-foe.ars, April 2007
17. “GUIMark Home << Craftymind”, http://www.craftymind.com/guimark/
18. “GUIMark Benchmark and Rendering Engine theory <<Craftymind”, http://www.craftymind.com/guimark-inside/
19. “Adobe Labs – Adobe Flash Player 10.1”, Adobe Labs, http://labs.adobe.com/technologies/flashplayer10/#FAQ, September 2010
20. Stevens T., “Internet Explorer falls below 50 percent global market share, Chrome usage triples”, engadget, http://www.engadget.com/2010/10/05/internet-explorer-falls-below-50-percent-global-marketshare-chr/, October 2010
21. StatCounter, “Top 5 Browsers from Sep 09 to Sep 10”, http://gs.statcounter.com/, Sept 2010
22. Harui A., “Threads in ActionScript 3”, http://blogs.adobe.com/aharui/2008/01/threads_in_actionscript_3.html, January 2008
23. theludditedeveloper, “Silverlight Security – Part One – Code obfuscation”, http://theludditedeveloper.wordpress.com/2009/07/26/silverlight-security-part-one-code-obfuscation/, July 2009
24. Twist J., “Securing Your Silverlight Applications”, MSDN Magazine, http://msdn.microsoft.com/en-us/magazine/ff646975.aspx, May 2010
25. Adobe, “Flex 3 – Adobe Flex 3 Help”, http://livedocs.adobe.com/flex/3/html/help.html?content=security2_01.html

26. Mills A. J.S., “ANTLR”, http://supportweb.cs.bham.ac.uk/docs/tutorials/docsystem/build/tutorials/antlr/antlr.html, University of Birmingham, 2005
27. Mossenbock H., Loberbauer M., WoB A., “The Compiler Generator Coco/R”, http://www.ssw.uni-linz.ac.at/coco/, University of Linz, January 2010