Accessibility in Rich Internet Applications

Damien P. Connaghan
School of Computing
Dublin Institute of Technology
Kevin St. Dublin 8, Ireland.

MSc Information Technology (Computing)

Dublin Institute of Technology, School of Computing
September 2008

Declaration

I herby certify that this dissertation which I now submit for assessment by the School of Computing, Dublin Institute of Technology on the programme of study leading to the award of ***MSc Information Technology (Computing)*** is entirely my own work and has not been submitted for assessment for any academic purpose other than in particular fulfilment for the stated above.

Signed

____
Your Name

Date

____

Abstract

Disabled computer users, such as blind users, interact with computers through special technologies called Assistive Technology. An example of an Assistive technology would be a screen reader, which describes the visible information on a computer screen in audio, to aid users with visual disabilities. There are many different types of assistive technologies to fit the various needs of disabled users.

In general accessibility describes the degree to which an object is accessible to as many people as possible. Similarly Web Accessibility describes the measure to which a web site is accessible to as many users as possible. Whether a user accessing a web page has a slow internet connection or is a disabled user using an AT, there are guidelines which web designers can follow to design accessible web pages. Existing guidelines, which have been successfully adopted within the Web community, include the World Wide Web Consortium's (W3C) Web Content Accessibility Guidelines (WCAG 1.0). Websites designed to WCAG 1.0 specification provide semantics in the webpage mark-up which ATs can interpret and successfully relay back to the user.

The term "Rich Internet Applications" (RIA) describes a dynamic web application. These dynamic web applications often have a strong emphasis on graphical representation and can be fond on almost all commercial websites today. An example of a RIA would be visually enhanced Web application embedded onto a web page, where portions of the web page change dynamically without interacting with the user.

Assistive Technologies relay the visible information from the screen back to the user on the initial loading of a webpage or at the user's request. However, should a RIA within a webpage change the visible information without reloading the page, the AT will be unaware of this change, unless notified. Since Assistive Technologies are not notified of changes within Rich Internet Applications, this makes the ATs last user notification of the webpage content obsolete and since the AT is unaware of the RIA modification, it will not notify the user of the change within the webpage.

This dissertation aims to explore the inadequate accessibility measures within Rich Internet Applications. It will also look at what Rich Internet Applications are, what is meant by accessibility today and how accessibility relates to Rich Internet Applications.

The dissertation also examines the new guidelines for developing Accessible Rich Internet Applications (WAI-ARIA 1.0). By implementing several Rich Internet Application technologies to the WAI-ARIA specifications, this dissertation will evaluate the ability of WAI-ARIA as a methodology for accessible Rich Internet Application design. The dissertation concludes by examining the reasons why existing Rich Internet Applications have inadequate accessibility measures, is it the fault of the developers or can Rich Internet Application vendors do more to make their platform more accessible?

Acknowledgements

First and foremost, I would like to acknowledge the support, encouragement and valued opinion of my supervisor Damian Gordon of the School of Computing at the Dublin Institute of Technology. I would also like to thank the course director Brian Duggan for the interesting seminars during the taught part of this course.

Thanks also to Debra Heeney of UCD for proofreading papers and drafts of this dissertation.

Table of Contents

Glossary of Acronyms Used

W3C	World Wide Web Consortium
WAI	World accessibility Initiative
WAI-ARIA	World accessibility Initiative- Accessible Rich Internet Applications
HTML	Hypertext Mark-up Language
XML	Extensible Mark-up Language
Web	World Wide Web
XSLT	Extensible Style sheet Language Transformation
ARPA	Advanced Research Projects Agency
AT	Assistive Technology

1. Introduction

1.1 Background

The term "Rich Internet Applications" (RIA) was first coined with the new age of the internet commonly referred to as Web 2.0 and is used to describe dynamic web applications. These web applications differ from traditional HTML-based web pages in that they have a strong user interaction and allow for an engaging user experience. Rich Internet Applications also have a strong emphasis on intuitive graphical interfaces which bring the user closer to the domain. Lawton (2008) argues that RIAs feature responsive user interfaces and interactive capabilities. They have many of the features and functionality of desktop software even though they run online.

This dissertation will provide an analysis of Rich Internet Applications. It will also investigate the semantics of Rich Internet Applications for aiding assistive technology. Currently Rich Internet Applications offers poor semantics for assistive technologies and this project will analyse how this can be improved. Many Rich Internet Applications are effectively multimedia movies, resulting in similar accessibility issues as video files.

Standard browser features that aid accessibility for typical web applications cannot be used with Rich Internet Applications. The main problem areas are:

- Browser options to adjust text size for visually impaired users.
- Applying customised style sheets to adjust colours (e.g. for people with dyslexia)
- Web browser bookmark capabilities, for easy page retrieval at a later date

The project will also discuss technological considerations, environmental factors, testing, cost and legal issues which may arise when delivering an accessible Rich Internet Applications.

1.2 Rich Internet Applications and Accessibility

Rich Internet Applications are still in their early stages, with very little published work relating to accessibility and usability in this context. The World Wide Web Consortium (W3C), who are the main international standards organisation for the World Wide Web released the 'Roadmap for Accessible Rich Internet Applications (WAI-ARIA Roadmap)' on the 4th February 2008. The roadmap lays out a vision of how Rich Internet Applications can become more accessible in the future.

Rich Internet Applications give online applications an interface similar to that of a desktop PC, offering extended functionality such as drag-and-drop, instantaneous processing and asynchronous feedback. All data required for an application to function is downloaded onto the users' computer and is processed locally. Initially, RIAs where thought to be run on proprietary sandboxes in the different user agents, as the development of HTML-based interfaces was reaching its limits. However, the introduction of the `XMLHttpRequest` object in the early 2000s and its rapid adoption by most of the browser vendors, originated a new breath of Web applications. The `XMLHttpRequest` Object defines an API that provides scripted client functionality for transferring data between a client and a server. By using `XMLHttpRequest` objects only parts of a web page need to be refreshed, increasing the application responsiveness and improving the user experience. The asynchronous nature of the transfer brings Web applications closer to desktop applications in terms of user interaction. (Velasco, *et al.*, 2008)

The W3C on its framework says there is no communication between assistive technologies and Rich Internet Applications, which could help build a bridge between the two technologies. *"Legacy GUI accessibility frameworks address these [accessibility] issues through a comprehensive accessibility application programming interface (API) and infrastructure to foster interoperability with assistive technologies. These APIs constitute a contract between applications and assistive technologies, such as screen readers, to enable them to access rich dynamic content with the appropriate semantics needed to produce a usable alternative. No such contract exists between modern Rich Internet Applications and assistive technologies, creating an accessibility gap for persons with disabilities."* (W3C, 2008)

The main American legislation that pertains to accessibility issues is covered under Section 508 of the Rehabilitation Act (1998). It requires that all federal agencies must act responsibly towards people with disabilities when introducing new technologies. *"Section 508 requires that Federal agencies' electronic and information technology is accessible to people with disabilities. IT Accessibility & Workforce Division, in the U.S. General Services Administration's Office of Government wide Policy, has been charged with the task of educating Federal employees and building the infrastructure necessary to support Section 508 implementation. Using this web site, Federal employees and the public can access resources for understanding and implementing the requirements of Section 508."*(Section 508 US Gov Accessibility Laws)

To improve the problem of inadequate accessibility standards, all American government websites must meet the terms laid out by this Act. Considering the poor accessibility levels offered by typical Rich Internet Applications such as Silverlight and Flash, it is unlikely that these applications will be adopted by the US government until significant improvements are made to them. Taking this into consideration, we can conclude that websites which must meet accessibility standards will benefit from the technological advantages offered by Rich Internet Applications, while the vast who adopt Rich Internet Applications will become more attractive, dynamic and interactive.

The Irish Disability Act was reformed in 2005 to accommodate the needs of the disabled in public services. It aims to ensure sensible system design so that the system is usable by all members of the community. The following extracts are relevant to Rich Internet Applications within the Republic of Ireland:

From Section 27: *"Where a service is provided to a public body, the head of the body shall ensure that the service is accessible to persons with disabilities".*

From Section 28: *"Where a public body communicates in electronic form with one or more persons, the head of the body shall ensure, that as far as practicable, the contents of the communication are accessible to persons with a visual impairment to whom adaptive technology is available"(*Irish Disability Act, 2005).

The Disability Act highlights how much importance the Irish government places on websites meeting accessibility standards. As Rich Internet Applications become more common, more Irish-based websites will fail to meet accessibility laws within Ireland. This gives added importance to improving accessibility semantics within Rich Internet Applications.

1.3 Project Aim

The principal aim of this project is to evaluate accessibility standards Rich Internet Applications. The completed thesis will further the academic understanding of accessibility measures in Rich Internet Applications. It will also identify criteria which can improve the semantics in Rich Internet Applications for assistive technologies.

The evaluation will cover a range of important aspects. The legal implications of a website not meeting the requirements laid down by the disability act will also be examined. W3C have recently laid out its framework for accessibility in Rich Internet Applications, this will be evaluated and a prototype will be developed along these guidelines.

Typical accessibility issues with Rich Internet Applications relate to browser issues, such as the inability to increase the font size for visually impaired. These usability issues prevent people with disabilities from accessing the website. Investigations will be carried out to identify solutions to make Rich Internet Applications friendlier to all sub groups.

Prototype Rich Internet Applications will be built using the most advanced accessibility features to determine how far behind Rich Internet Applications are in terms of international accessibility laws.

1.4 Project Objectives

1 Evaluate the inadequate accessibility offerings of Rich Internet Applications

While the graphical enhancements offered by Rich Internet Applications are enormous, considerations need to be given to the accessibility for people with disabilities. This is a major problem with current Rich Internet Applications, and is not very well researched. The objective is to explore and evaluate methods of achieving accessibility standards as laid out by current international laws.

2 Design and Develop Rich Internet Applications Semantics for Assistive Technologies

To demonstrate the accessibility shortcomings of Rich Internet Applications and also to develop suitable accessibility framework, a Rich Internet Applications prototype will be developed as part of this project.

3 Screen Reader Software Integration.

Screen readers are widely used by visually impaired users to read aloud everything on computer screens, including text, pull down menus, icons, dialog boxes and websites. With Flash or Silverlight, there is a problem making Rich Internet Applications meet current accessibility standards. The project prototype will be assessed with an assistive technology.

4 Evaluate the social, legal implications of inaccessible Rich Internet Applications

The website audience will vary in user characteristics. This objective is to diagnose what makes a website suitable for all disciplines. It is expected that the attitude towards a rich content website will differ. A Rich Internet Application has a more sophisticated look and feel to a traditional website and the project aims to underline the end users need for education on website interaction, where features are not self explanatory.

1.5 Thesis Roadmap

Chapter 2 Rich Internet Applications

Examine the existing knowledge in the field of Rich Internet Applications. This chapter starts by discussing basic Web technologies such as the Internet and how it was first constructed. Web 2.0 technologies are explored, of which Rich Internet Applications are a subset. Various types of Rich Internet Application technologies are examined

Chapter 3 Accessibility

The chapter starts by examining various trends of Web design such as Universal Design and Inclusive Design. Assistive Technologies are then explained, before examining legislation relating to disabled users. This chapter closes by giving the reader a brief overview of WAI-ARIA, which is the Accessible Rich Internet Application framework proposed by the World Wide Web Consortium.

Chapter 4 Accessible Rich Internet Applications

Accessible Rich Internet Applications is an intersection of chapters 2 and 3, discussing Accessible Rich Internet Applications. Published guidelines for achieving Accessible Rich Internet Applications are examined such as the Accessibility Interoperability Alliance and the Web Accessibility Initiative's much talked about WAI-ARIA 1.0.

Chapter 5 Web Accessibility Initiative - Accessible Rich Internet Applications 1.0 (WAI-ARIA)

This chapter digs into the proposals of WAI-ARIA, examining the key recommendations and how an application would be designed using the principles of WAI-ARIA.

Chapter 6 Research Methods

An evaluation of how to approach application design using WAI-ARIA. Research into Microsoft Silverlight and Macromedia Flash, which are two leading technologies for building Rich Internet Applications are conducted here.

Chapter 7 Implementation of Prototype Applications

This chapter is concerned with documenting the software development of the two prototype applications. The specific software development methodologies used are explained. Explanations into how the accessibility features where implemented can be found in this chapter.

Chapter 8 Evaluation

In this chapter the software is evaluated to assess if adequate accessibility measures were achieved in each prototype application. An evaluation of the level of difficulty in implementing software with accessibility features is conducted.

Chapter 9 Conclusions and Future Work

The final chapter looks at the future work which needs to be taken in order to achieve accessibility in Rich Internet Applications. Final thoughts on where Web accessibility is and where it is going are given, before suggesting a few possible inclusions into WAI-ARIA 2.0

2. Rich Internet Applications

2.1 Introduction

This chapter first introduces research in the area Rich Internet Applications. This chapter begins with basic Web technologies and goes on to introduce Rich Internet Applications. Descriptions of the leading Rich Internet Applications technologies are discussed and the chapter closes by discussing Rich Internet Applications development methodologies.

2.2 Web 1.0

This section discusses the key concepts in Web 1.0 technologies. Web 1.0 would be the first generation of Web technologies such as basic text and image web sites. Later Web technologies provide dynamic features, which early websites lacked.

2.2.1 The Internet

To describe the birth of the Internet, one needs to go back to the 1950's around the time of the Space Race. In 1957 the USSR launched Sputnik, the first artificial earth satellite. In response to Sputnik, in 1958, the USA issued directive 5105.15 establishing the Advanced Research Projects Agency (ARPA). The organisation produced the first U.S. satellite within 18 months. Within a few years of being established ARPA focused on computing networking and communications technology. O' Neill (1995) agues that the first order of business at ARPA was to broaden the scope of the office from existing command and control research to include more general aspects of information processing.

In 1962, Dr J.C.R. Licklider became the head of ARPA's research aimed at improving the militaries use of computer technology. Licklider was a visionary, who saw quick results through the expansion of ARPA's contracts from the private sector to the universities. This laid the foundations for what became ARPANET, the world's first operational packet switching network in 1972. Creating a network of computers was part of Licklider's desire to enhance the environment for conducting computer research. In 1960 Licklider articulated his vision of the future of the human-computer interaction in an influential article entitled "Man-Computer Symbiosis." (O'Neill, 1995)

From this first network, the internet has grown into a global network where devices can communicate through recognised protocols. Information can be exchanged between devices via the internet. The internet is the basic foundation to what is commonly referred to as the World Wide Web. Mercier *et al. (*1998) describes The Internet is not simply a single large computer network that spans the globe. The

Internet is a collection of thousands of individual networks-some publicly supported, some privately owned-connected by dedicated, special-purpose computers called routers. The owners of each of these individual networks connect them to the Internet either as a matter of public policy or because they believe there are long-term business advantages to Internet participation.

2.2.2 World Wide Web

The World Wide Web is an interlinked connection of hypertext documents which are accessible over the internet. To access hypertext documents users can use one of any number of web browsers to browse text, images and other multimedia sources. The World Wide Web was invented by Tim Burners-Lee in 1992. Begun in 1989 in Switzerland, the WWW or web was created to solve a very specific problem: how could researchers at CERN share their scientific data with others whose computer, network, and operating system were unknown? In other words, how could text, data, and pictures be represented and transmitted in a common format so that they could be viewed by a Cray running UNIX and by a PC running Windows, without maintaining separate copies of the file for each potential type of viewer? The answer was to configure a generalized system of servers and viewers such that they would run on any computing platform. (Weaver 1997)

Viewing a hypertext document (web page) usually involves typing in a URL (Unified Resource Locator) into a web browser. The web browser uses a protocol called HTTP (Hypertext Transfer Protocol) to request a web page from the web server which serves the URL.

There are several dominant Web browsers available for download today. Microsoft ships its Operating Systems with its own Web browser called "*Internet Explorer*". Mozilla Firefox is also a very popular browser and it is a popular choice among Web Developers who create their own additional features for Firefox to extend the functionality. The most recent Web browser to be released in 2008 is "*Google Chrome*", which is expected to become very popular given Google's dominant presence in the Web.

2.2.3 HTML

Hypertext Mark-up Language is the predominant mark-up language for web sites. It provides a means to describe the layout and properties of textual and imagery information in a document. HTML is written in a specified format which uses angle brackets to declare the appearance of a page through predefined HTML elements such as image tags or font tags. (Korpela, 1998)

Tim Burners-Lee first introduced HTML tags in 1991 which described 22 elements that made up the entire HTML specification at this time. As the World Wide Web grew, so did the required HTML specification and today HTML 4.01 is the standard on the web. W3C have just released a working draft of HTML 5.0 but this specification is not expected to be ready for at least 5 years. (W3C HTML 5, 2008)

```html
<html>
  <head>
    <title>Tutorial: HelloWorld</title>
  </head>
  <body>
    <h1>HelloWorld Sample</h1>
  </body>
</html>
```

Figure 2.1 Sample HTML code

2.2.4 Traditional Web Applications

Traditional Web Applications are Web-based applications which are primarily built on mark-up technologies (such as HTML and XHTML) and exhibit the basic characteristics and functionality of a website. Characteristics such as forms, Web controls such as checkboxes, drop down lists would be akin to a traditional Web application.

In a "traditional" Web application the HTML interface is computed by the server at each user's request. When the user interacts with a link or a button on the page the server is invoked and a new page is computed and sent back to the client. The role of the browser is simply to intercept the user's actions, deliver the request to the server, and display a whole new interface even if the change is minimal (Carughi 2007).

2.2.5 Extensible Mark-up Language (XML)

The Extensible Mark-up Language (XML) is an all purpose mark-up language which allows users to define their own elements within the document. The most common purpose of XML is for sharing structured data across multiple information systems. XML is used everywhere on the World Wide Web to communicate information because it is a simple, easy to understand fee-free specification. XML documents must be well formed, meaning if an element is used it must have a corresponding closing bracket in the correct position in the hierarchy of the document.

```
<deeptreeconfig sLeftFrameHeight="84">
      <TopXMLSrc>navtree/avms.xml</TopXMLSrc>
      <TreeLabel></TreeLabel>
      <StartPage>blank.html</StartPage>
      <Locale>en-us</Locale>
      <LocaleTextDirection>LTR</LocaleTextDirection>
</deeptreeconfig>
```

Figure 2.2. Sample XML code

The widespread employment of XML requires the development of efficient methods for manipulating XML data. Query languages, such as XQuery and XPath, take into consideration the inherent structure of the data and enable querying both on its structure and on simple values. (Mirella *et al*., 2005)

2.3 Web 2.0

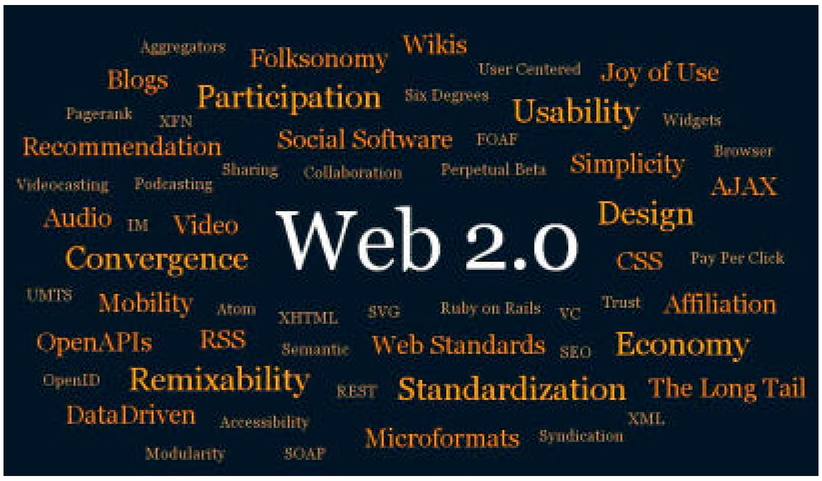


Figure 2.3 A Tag Cloud representing the main themes of Web 2.0 (Angermeier, 2005)
Markus Angermeier : Web 2.0 Mindmap Translated versions
http://kosmar.de/archives/2005/11/11/the-huge-cloud-lens-bubble-map-web20/

The concept of "Web 2.0" began with a conference brainstorming session between O'Reilly Inc. and MediaLive International. Dale Dougherty, web pioneer and O'Reilly VP, noted that far from having "crashed", the web was more important than ever, with exciting new applications and sites popping up with surprising regularity. (O’Reilly 2005)

O’Reilly (2005) summarises what he believes to be the core competencies of Web 2.0 companies.

? Services, not packaged software, with cost-effective scalability

? Control over unique, hard-to-recreate data sources that get richer as more people use them

? Trusting users as co-developers

? Harnessing collective intelligence

? Leveraging the long tail through customer self-service
? Software above the level of a single device
? Lightweight user interfaces, development models, AND business models

Although the term Web 2.0. has been widely used as a marketing tool, this term still describes a new wave of Web technologies which have emerged with more superior functionality to traditional Web applications. One of the most significant of these new features is Rich Internet Applications and the growing list of development environments which can be used to develop Rich Internet Applications further underline the increasing growth of Rich Internet Applications.

2.3.1 Necessity for Richer Internet Applications

Currently, the complexity of tasks performed though Web applications is increasing, this is particularly true when high levels of interaction, client-side processing and multimedia capacitates have to be provided. In this context traditional HTTP-HTML Web applications are showing their limits and developers are building the future of the Web using Rich Internet Applications technologies, which are Web applications with many additional features. (Preciado *et al*., 2007)

Preciado *et al* (2005) outlines that traditional Web applications are inadequate to support the interaction and presentation functionalities demanded by the users. Traditional Web application methodologies and tools are incomplete or inadequate to answer the new functionalities that users demand from Web applications, such as, for example, effective integration of audio and video and high level of interactivity. The use of Rich Internet Applications answers such needs through the introduction of rich contents that can be delivered on different devices. At the one hand, Rich Internet Applications allow embedding audio, video and other media contents under one plug-in installation (such as Flash Player or Java). On the other hand, Rich Internet Applications guarantee high performance levels for any kind of data by avoiding continuous page refreshment. Preciado *et al.* (2005) describes the four main problems with traditional Web applications.

(a) **Process Problems:**	Complex Web applications often require that the user navigates through a series of pages to complete a single task (e.g. the task of booking a flight).
(b) **Data Problems:**	They do not support interactive explorations of the data. Usually, the user has to search data through the use of input forms and then to navigate the hypertext to see the desired data. Different data visualization and interactive manipulation in an effective way would reduce the complexity of the data shown to the user.
(c) **Configuration Problems:**	Many Web applications require the configuration of a product/system from multi criteria choices, but are, in general, unable to present the

	customised product/system to the user in an intuitive way and in a single step.
(d) **Feedback Problems:**	They do not allow a continued and ordered interaction without page refreshments, so the interaction of the user with traditional Web pages is quite limited.

Figure 2.4. Preciado *et al*. (2005) describes the four main problems with traditional Web applications.

Bozzon. *et al* (2007) describes the increasing complexity of Web applications in that current Web technologies are starting to show usability and interactivity limits. The user experience in thin-client Web applications is not comparable to desktop interfaces, responsiveness is lower due to network overhead and unnecessary round-trip server access, and disconnected usage is not supported. Rich Internet Applications have been recently proposed as the response to the abovementioned drawbacks. They provide sophisticated interfaces for representing complex processes and data, while minimizing client-server data transfers and moving the interaction and presentation layers from the server to the client. In the case of why Rich Internet Applications have emerged, it is clear that necessity was the mother of invention. Traditional applications lacked the tools to develop rich user experiences and could not cope with the increased demands for high levels of user interaction and desktop like features in a web site. These requirements opened the door for Rich Internet Applications, which have merged desktop-like features to the Web browser.

2.3.2 The Arrival of Rich Internet Applications

Rich Internet Applications have been recently proposed as the response to usability and interactivity limits on the World Wide Web. They are a variant of Web-based systems providing sophisticated interfaces for representing complex processes and data, minimizing client-server data transfers and moving the interaction and presentation layers from the server to the client. Typically, a Rich Internet Application is loaded by the client along with some initial data; then, it manages data rendering and event processing, communicating with the server when the user requires further information or must submit data (Bozzon *et al*., 2006)

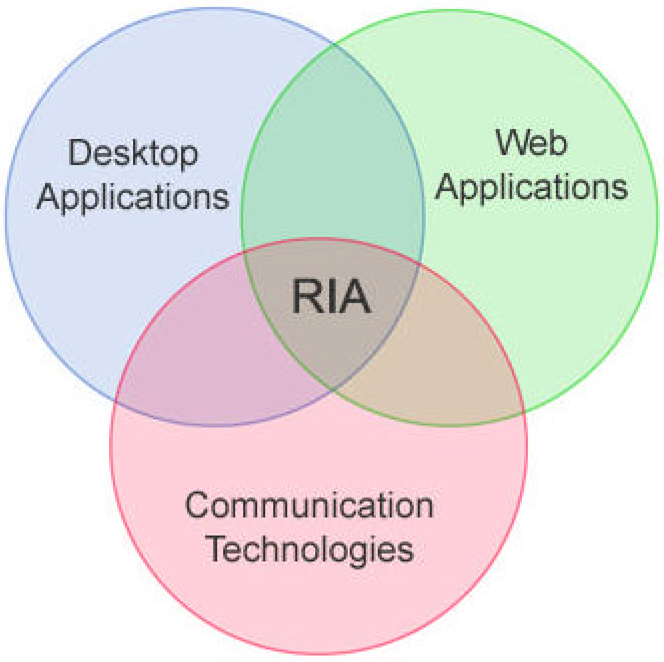


Figure 2.5 A Venn Diagram of RIA (Whatley, 2007)
Simon Whatley : Rich Internet Applications - A Background
http://www.simonwhatley.co.uk/tag/silverlight accessed 11/09/2008

Rich Internet Applications are gaining popularity thanks to the facilities they provide to develop Web applications with multimedia, high levels of interactivity, collaborative work, and/or homogeneous presentation requirements at the client side. However, this new kind of Web application currently lacks complete methodologies and models which aid its design and development (Preciado, *et al*., 2007).

Rich Internet Applications offer online and offline capabilities, sophisticated user interfaces, the possibility to store and process data directly on the client-side; they offer high levels of user interaction, usability and personalisation, minimise bandwidth usage, and separate presentation and content at the client-side (Preciado, *et al*., 2007).

Preciado *et al*. (2005) defines Rich Internet Applications as the fusion of the interactive and multimedia user interface functionality of desktop applications with the Web applications. Preciado *et al*. however, does capture the true essence of what a Rich Internet Applications should be where it points out that a "*rich internet application is desktop functionality brought to the web browser, allowing for a more natural user experience*".

Rich Internet Applications seem to be the solution offered by computer scientists for the more demanding users. Unlike the previous generation of Web applications, Rich Internet Applications plan the leverage of the user experience with more powerful graphical user interfaces (GUI) including charts, drag & drop, etc. Rich Internet Applications also aim to remain as platform-independent and as setup-free as possible. One should be able to work on data and tools both available online and these tools must be usable without setting up or configuration on the client computer. However, Rich Internet

Applications also should be more than a few good looking graphical user interfaces because the desktop applications have made the users accustomed to a certain level of responsiveness and customisability (Jadoul, *et al*., 2006). Rich Internet Applications do not try to replace HTML, because this language still remains ideal for representing simple data (including graphs and photos) without high interaction levels (Morales-Chalarro, *et al*., 2008).

Carughi (2007) argues that Rich Internet Applications extend the traditional Web architecture by moving part of the application data and computation logic from the server to the client. The aim is to provide more reactive user interfaces by bringing the application controller closer to the final user, minimizing server round-trips and full page refreshes. In Rich Internet Applications instead, the client-side logic handles each user interaction updating interface sub-elements only. Furthermore, when client-server communication is needed to transfer some data, it can be performed in the background, asynchronously, allowing continuous user interaction with the interface. This, together with HTTP/1.1 persistent connections, are the key ingredients of a technique called HTTP trickling, one of the solutions enabling servers to initiate server-to-client communication (server-push) once a first HTTP request is performed.

Through Rich Internet Applications, the client's memory is available to be used by the application; the amount of client storage capacity depends on the selected Rich Internet Applications technology or on the preferences. In a Rich Internet Application it is common to store persistent and volatile content also on the client (e.g., a shopping cart on an e-commerce application or a calendar of appointments could be stored locally on the client), to manipulate it (e.g. add, delete or modify ordered items or appointments) and to send the manipulated content to the server once the specified whole operation has been completed (Preciado *et al*., 2007).

2.3.3 Rich Internet Applications Architecture

Urbieta *et al* (2007) describe the many features which characterise Rich Internet Applications from a Web engineering point of view such as rich interaction capabilities, complex client-side processing, and the elimination of full page refreshing to provide navigation and multimedia animations.

In the following sections the different components of a RIA will be discussed. These are the Business Logic Foundations, the Presentation Foundations and the Communication Foundations.

2.3.4 Business Logic Foundations

In traditional Web applications processes are executed only on the server: the client performs a request and the server builds a new page that is sent to the client as a response. Rich Internet Applications have a different navigation structure to traditional Web applications. Preciado *et al.* (2007) explains that Rich Internet Applications operate as a single page application where nested pages are able to be processed by the client or by the server. Due to the augmented process capability of the client, in Rich Internet Applications both the client and the server can carry out complex operations (e.g. data filtering, numeric operations, etc.). The complexities of the tasks that can be performed in a Rich Internet Applications require a richer kind of distributed functionality with respect to traditional Web applications. Moreover, additional mechanisms are needed to start a conversation (request) from both sides (client and server).

Rich Internet Applications allow distribution of the business logic. This situation does not solely include the distribution between client and server tasks, but also a new kind of task called "*mixed*".

The business logic can be therefore performed in three ways.

(a) **Client-Side**: When all the logic required for a particular task is performed only on the client.

(b) **Server-Side**: When all the logic required for a particular task is performed only on the server.

(c) **Mixed**: When the logic is distributed between the client and the server, in such a way to complete a complex task, one part is carried out on the client and another on the server.

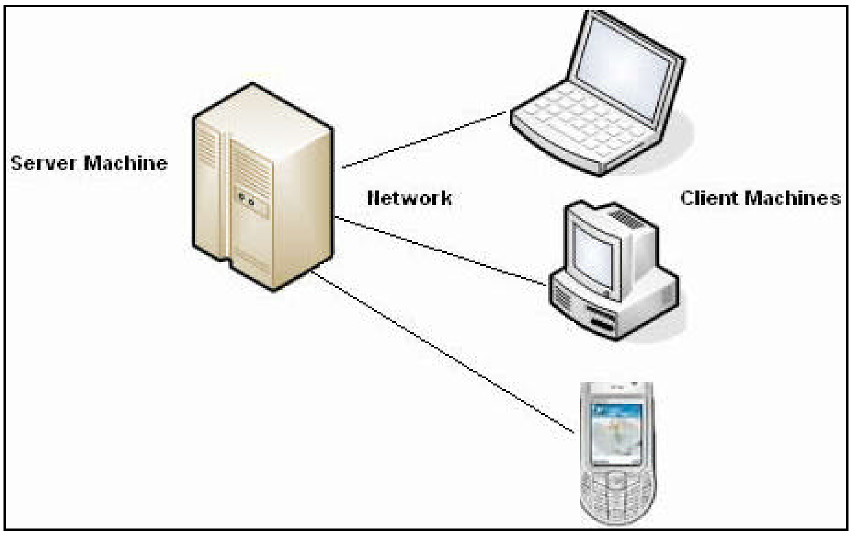


Figure 2.6 Client Server Architecture
www.acm.org accessed 11/09/2008

2.3.5 Presentation Foundation

Presentation in traditional web applications is quite limited. On the one hand, traditional Web application renders are not able to provide multimedia native support and they need plug-ins in order to be able to show video and audio homogeneously at the client-side (e.g., to show the content of the www.youtube.com site). However, they do not approach rich interactions (e.g., drag and drop) and animations: to obtain them expensive JavaScript browser-dependent code needs to be written. Rich Internet Applications offer new functionalities that improve presentation and users interactions.

Presentation in Rich Internet Applications focuses on the layout, on the definition of styles, and on capturing the behaviours of the application and the interaction of the users. However, for managing the different aspects, Rich Internet Application UIs are very dependent on the device where they are going to be rendered. The device establishes constraints like screen size and multimedia support and this should be taken into account to adapt the UI for achieving better user experiences (Preciado *et al.*, 2007). Although Rich Internet Applications interfaces aim at resembling traditional desktop applications, Rich Internet Applications are generally composed of task-specific client-run pages deeply integrated with the architecture of a traditional web application. (Carughi 2007)

2.3.6 Communication Foundation

Traditional Web applications allow synchronous connections and the communications are originated at the client-side. Rich Internet Applications allow both synchronous and asynchronous communications. Distribution of data and functionality across client and server broadens the features of the produced events as they can originate, be detected, notified and processed in a variety of ways. Communication is a cross-cutting concern related to data synchronisation, business logic distribution and presentation experiences (Preciado *et al*., 2007).

2.3.7 Types of Rich Internet Applications

The following is a list of Rich Internet Application technologies which can be used to develop Rich Internet Applications. This list is arranged in no particular order as each technology in principle offers similar outputs.

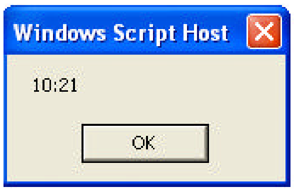
- ***JavaScript / Ajax***

JavaScript was the first client side scripting language and it can be interpreted by all web browsers.
It was released in 1995 by Netscape and Sun, JavaScript interacts with HTML code and makes Web pages and Ajax applications more active (Paulson 2005). Asynchronous JavaScript and XML (AJAX) is a combination of techniques which can be used to process asynchronous calls a web application allowing the application to achieve greater real time interoperability. Ajax is the technology used by Google for projects such as Gmail and Google Maps. Several open source Ajax Frameworks have been developed to simplify the process.

Sample Javascript

Executing the following code in your web page would extract the hour from the variable called str and display it in a pop up dialog window, as show below.

```
var  str, rarr;
str= "Thursday, August 24, 2008 10:21 AM";

WScript.Echo(ExtractDatePartfromString(str, "hour"));

function ExtractDatePartfromString(str, part)
{
  var rarr, re;
  if(part == "hour")
            re = /\d\d:\d\d|\d:\d\d/gim;

  rarr = str.match(re);
  return rarr[0];
}
```


Windows Script Host

10:21

Figure 2.7 Sample JavaScript code and the resulting dialog

- ***Google's GWT framework***

The 'Google Web Toolkit' released in 2006 allows the development and testing of JavaScript based AJAX Rich Internet Applications using the Java language. Designed specifically for Java developers, GWT enables Java programming, refactoring, debugging and unit testing of Rich Internet Applications using existing tools (e.g. Eclipse).

GWT is an AJAX framework, developed by Google, which permits us to create RIAs by writing the browser-side code in Java, thus gaining all the advantages of Java (e.g. compiling, debugging, etc.) and generating a generic Javascript and HTML code that can be executed in any browser. Moreover, GWT makes every attempt to be flexible allowing us to integrate with other client AJAX frameworks (e.g. Script.aculo.us, Dojo, Yahoo! UI, and so on) and with server Java frameworks such as Struts, EJB, etc (Meliá *et al*., 2008).

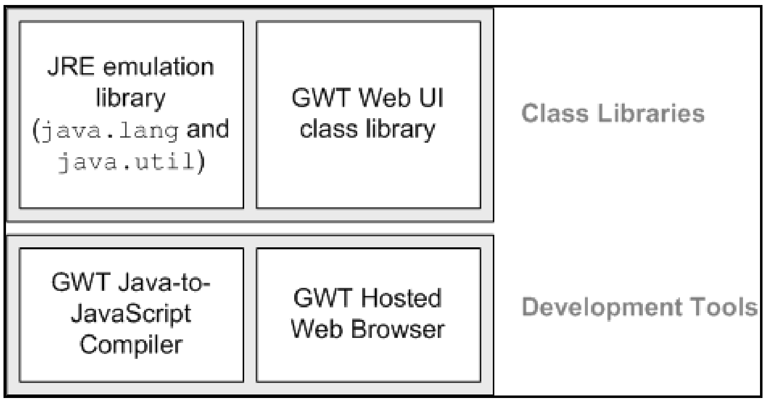

JRE emulation library (java.lang and java.util)
GWT Web UI class library
Class Libraries
GWT Java-to-JavaScript Compiler
GWT Hosted Web Browser
Development Tools

Figure 2.8 GWK Framework (Lumb, 2008)
http://ianlumb.wordpress.com/2007/01/06/google-office-for-the-blackberry-coming-soon/ accessed 11/09/2008

- ***Adobe Flash, Adobe Flex and Adobe AIR***

A powerful cross-platform technology, which can create an application UI. Adobe Flex provides the option to create Flash user interface by compiling MXML, an XML based interface description language which is allows the developer to define elements of the UI. Developed by Macromedia and now owned by Adobe, executed using the Flash browser plug-in and written in a language called ActionScript; the plug-in is very widespread (Lammarsh *et al*., 2008).

Adobe Flex is written entirely in ActionScript 3, which was introduced as part of the Flash 9 Player plug-in. Flex applications are deployed as compiled byte code that is executed within the Flash Player runtime system. The core of Flex is the developer-centric Flex framework, a library of ActionScript 3 objects that provide a great foundation for building rich internet applications. Writing applications with Flex is similar to developing in .NET or Java (Mertens *et al.*, 2008).

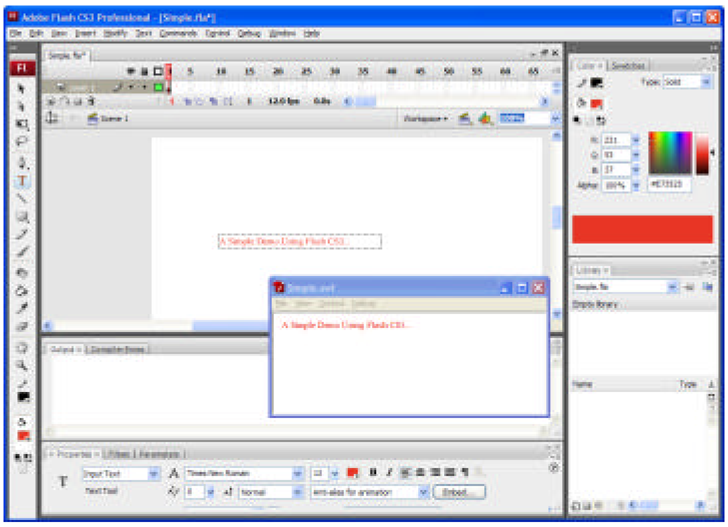
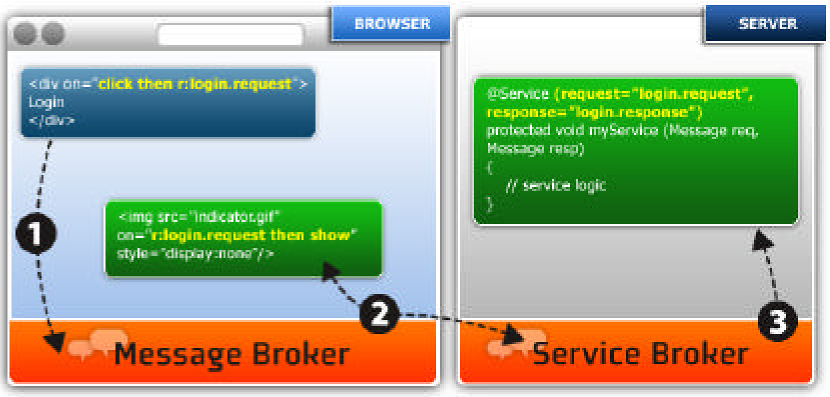

Adobe Flash CS3 Professional - [Simple.fla*]
A Simple Demo Using Flash CS3.
Simple.swf
A Simple Demo Using Flash CS3.

Figure 2.9 Scrennshot of Adobe Flash (Infoworld, 2007)
James R. Borck : Adobe Flex Builder speeds RIA development
http://www.infoworld.com/article/08/04/24/17TC-adobe-flex-builder_1.html?source=fssr accessed 11/09/2008

• *Appcelerator*

Appcelerator is an open source platform for developing Rich Internet Applications using a service-oriented architecture. This technology allows interoperability with standards such as HTML, CSS and JavaScript.



```
<div on="click then r:login.request">
Login
</div>
```

```
<img src="indicator.gif"
on="r:login.request then show"
style="display:none"/>
```

```
@Service (request="login.request",
response="login.response")
protected void myService (Message req,
Message resp)
{
   // service logic
}
```

Figure 2.10 The appcelerator framework (Appcelerator, 2008)
http://www.appcelerator.com/product_details.html accessed 11/09/2008

Web Expression Language: Instead of using highly complicated functions and commands, Appcelerator eases everything up by using the most basic in web development: HTML.

Widget Framework: Appcelerator uses the widget technology in building their application. Instead of using a very complicated full blown web development technique, widget framework eases the pressure of complication for developers.

Message Broker: HTML, by itself is a very lightweight message transmitting technique. However, this is only used within the server. Appcelerator on the other hand has found a way to establish a faster response for the application. Called the message broker, the client side messaging eases the web expression language since it is fully introduced to system.

- ***OpenLaszlo***

Developed by Laszlo Systems Inc, OpenLaszlo is a powerful open source framework for developing Rich Internet Applications. The OpenLaszlo server compiles programs written in the LZX language (a mixture of XML tags and JavaScript) into either AJAX or Adobe Flash byte code, currently supporting Flash7 and Flash8.

OpenLaszlo consists of Java libraries that generate Flash or DHTML applications from XML and JavaScript source files. For environments having Web browsers with Flash Player or plug-in OpenLaszlo applications can be deployed as Flash files. For platforms without Flash capabilities applications can still be deployed with DHTML interface similar to Flash version. (Rinc 2008)

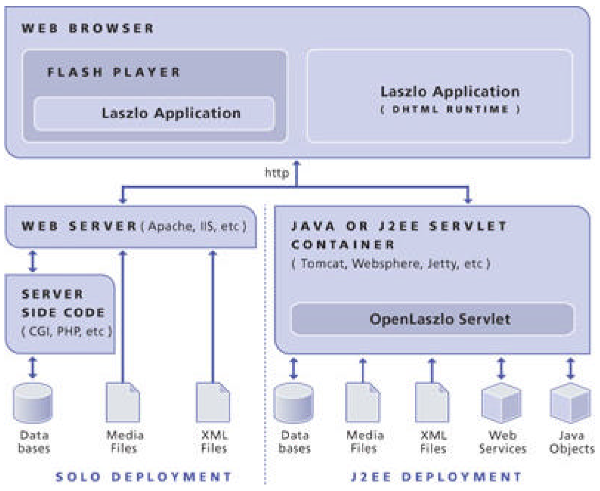


Figure 2.11 OPen Laszlo Framework Casario, 2008)
Marco Casario : Released OpenLaszlo 3.0
http://casario.blogs.com/mmworld/2005/04/released_openla.html accessed 12/09/2008

- ***Curl 5.0, REBOL 2.6 and Seaside for Smalltalk***

These are three alternatives for using Java to develop Rich Internet Applications. Curl operates by using Client-side persistent data. While REBOL does not require a browser and Seaside for Smalltalk uses a minor extension to Smalltalk to provide a much richer web experience.

The full Curl platform consists of three components:

An extensible programming language: (the Curl content language) designed to create interactive Web content.

Runtime engine: (the Curl RTE) that executes Curl programs and renders the resulting content.

IDE: (integrated development environment) to construct rich Internet client-side applications.

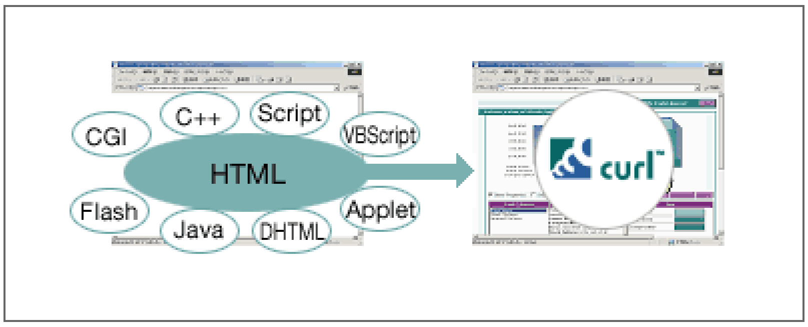
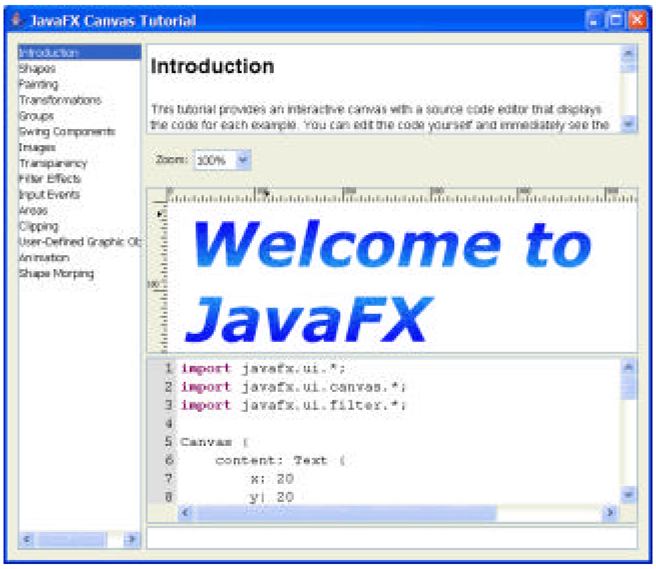


Figure 2.12 Introducing Curl Framework (Curl, 2008)
http://www.curl.com/products/feature/ accessed 10/09/2008

- ***JavaFX***

Sun Microsystems recently announced the Java FX Script Programming Language. "*The JavaFX Script™ language is a declarative and statically typed programming language. It has first-class functions, declarative syntax, list-comprehensions, and incremental dependency-based evaluation. The JavaFX Script language makes intensive use of the Java2D swing GUI components and allows for easy creation of GUIs.*" It shares M's goal of write-once-run-anywhere but does not duplicate M's on-phone programming capabilities (Cook 2007).

Designed to provide a consistent experience across a wide variety of devices including desktops, (as applets and stand-alone clients) set-top boxes, mobile devices, and Blu-Ray players. JavaFX Script can develop rich 2D interfaces using a declarative syntax similar to XAML.



Introduction

This tutorial provides an interactive canvas with a source code editor that displays the code for each example. You can edit the code yourself and immediately see the

```
import javafx.ui.*;
import javafx.ui.canvas.*;
import javafx.ui.filter.*;

Canvas {
    content: Text {
        x: 20
        y: 20
```

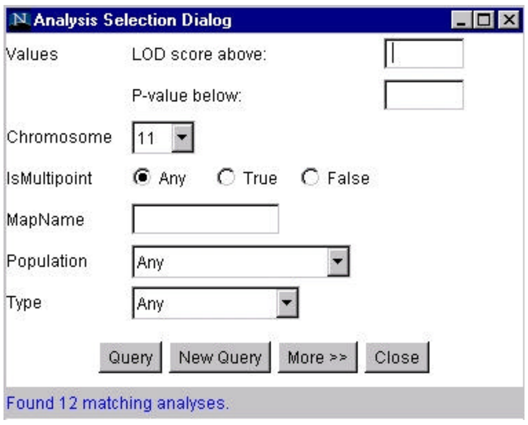
Figure 2.13 Screenshot of JavaFX development environment (Weaver, 2007)
James Weaver : Learning the Fundamentals of the JavaFX Script Language
http://www.curl.com/products/feature/ accessed 19/09/2008

- ***Java applets***

Java applets run in standard HTML web pages and can start automatically when their web page is opened with a modern web browser.

When opened by a web browser, Java applets can access the clients screen, as well as the speakers, keyboard and mouse, as well as access to the Internet, which can be used to develop highly interactive desktop like applications inside the web browser.

Developed by Sun, executed using the Java Virtual Machine (VM) and written in the Java language; many free libraries are available for Java and the VM is widespread. (Lammarsh *et al.*, 2008)



Values	LOD score above:	
	P-value below:	
Chromosome	11	
IsMultipoint	[✓ Any] True False	
MapName		
Population	Any	
Type	Any	

Found 12 matching analyses.

Figure 2.14 Screenshot of JavaFX development environment (Chines , 2002)

Peter Chines : How Analysts Use SIFTER
http://www.hps.com/~tpg/toolbox/sifter/intro/users.html accessed 13/09/2008

• *Java applications*

Java Rich Internet Applications can be developed which will be run in the browser or as free standing applications via Java Web Start which integrates with the desktop. Java Rich Internet Applications can utilise the full Java libraries to deliver interactive and dynamic user experiences. Typical frameworks used for developing Java Rich Internet Applications include XML-based XUI, Swixml, or Canoo's, UltraLightClient.

The Java programming language that evolved out of a research project started by Sun Microsystems in 1990 (Arnold & Gosling 1996; Gosling *et al*. 1996) is one of the most exciting technical developments in recent years. Java combines several features found in different programming paradigms into one language. Features such as platform independence for portability, an object-orientation model, support for multithreading, support for distributed programming, and automatic garbage collection, make Java very appealing to program developers. Java's "*write-once, run anywhere*" philosophy captures much of what developers been looking for in a programming language in terms of application portability, robustness, and security. The Java's flexibility, however, is its performance due to the high degree hardware abstraction it offers. (Iffat *et al*. 2000)

Sample Java Application

```java
private void credit(APDU apdu) {

   if ( ! pin.isValidated() )
     ISOException.throwIt(
        SW_PIN_VERIFICATION_REQUIRED);

    byte[] buffer = apdu.getBuffer();
    byte numBytes = buffer[ISO7816.OFFSET_LC];
    byte byteRead =
              (byte)(apdu.setIncomingAndReceive());

    if ( ( numBytes != 1 ) || (byteRead != 1) )
     ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
     byte creditAmount = buffer[ISO7816.OFFSET_CDATA];

  }
```

Figure 2.15 Sample Java code

- ***Microsoft Silverlight***

Microsoft Silverlight, which can be considered a subset of Windows Presentation Foundation (WPF), allows developers to develop Rich Internet Applications using Visual Studio. Silverlight 2 Beta 2 is currently the latest release in this new Rich Internet Applications technology. (MS Silverlight 2008) Silverlight's greatest asset is the ability to integrate C# within the applications, resulting in powerful applications which can be easily developed by the `.Net development community`. Client machines need to install a small (about 2MB) plug-in (Silverlight Runtime) in order to be able to play Silverlight contents. Developed by Microsoft, executed using the Silverlight plug-in which is a smaller version of the `.NET framework` and written in various languages supported by `.NET`, particularly C# and Visual Basic; the `.NET framework` is very powerful and has spread in the commercial area very quickly. (Lammarsh *et al*., 2008)

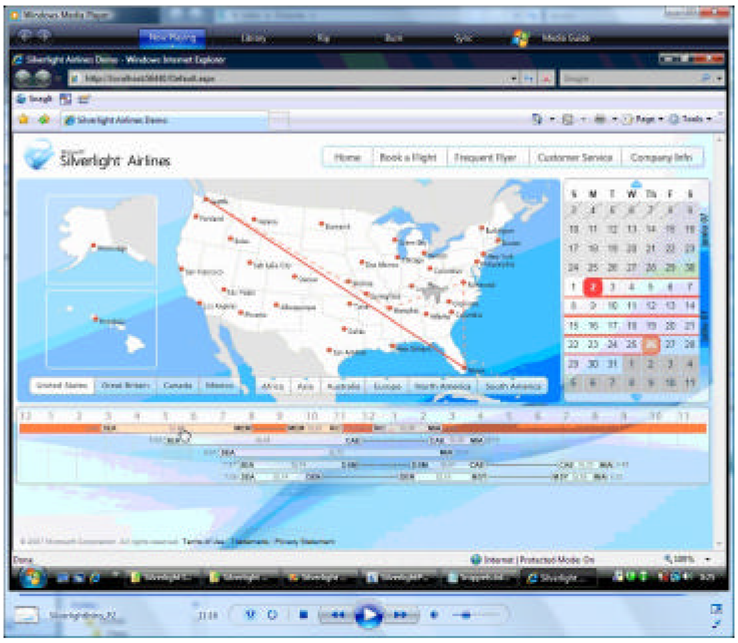


Figure 2.16 Sample Silverlight Flight booking System (Silverlight Team , 2008)
www.silverlight.net accessed 13/09/2008

- ***Mozilla Prism***

Mozilla Prism is a new product in development which integrates web applications with the desktop, allowing web applications to be launched from the desktop and configured independently of the default web browser. It has been used by Google Docs and Gmail along with AJAX. This would allow users to interact with their Web applications in an offline or online mode, giving Web applications a connectionless feel.

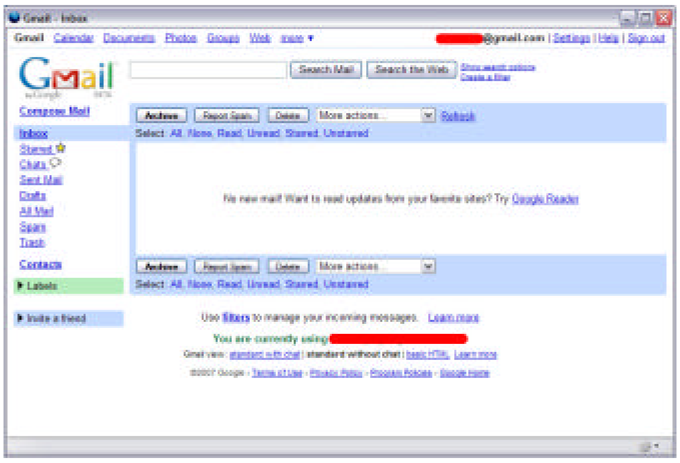
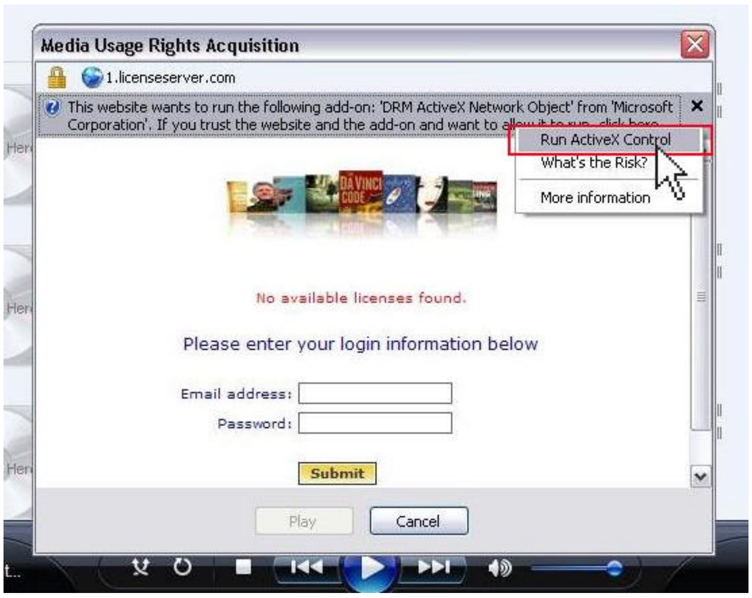

Gmail - Inbox
Gmail Calendar Documents Photos Groups Web more ▾
[redacted]@gmail.com | Settings | Help | Sign out
Gmail
Search Mail
Search the Web
Show search options
Create a filter
Compose Mail
Inbox
Starred
Chats
Sent Mail
Drafts
All Mail
Spam
Trash
Contacts
Labels
Invite a friend
Archive
Report Spam
Delete
More actions...
Refresh
Select: All, None, Read, Unread, Starred, Unstarred
No new mail! Want to read updates from your favorite sites? Try Google Reader
Use filters to manage your incoming messages. Learn more
You are currently using [redacted]
Gmail view: standard with chat | standard without chat | basic HTML Learn more
©2007 Google - Terms of Use - Privacy Policy - Program Policies - Google Home

Figure 2.17 Gmail system which is built with Mozilla Prism (Lynch , 2008)Mark Lynch : Gmail, Docs, Calendar and Analytics standalone with Prism http://www.lynchconsulting.com.au/blog/index.cfm/2008/3/24/Gmail-Docs-Calendar-and-Analytics-standalone-with-Prism accessed 13/09/2008

• *ActiveX controls*

ActiveX objects are embedded in HTML to develop Rich Internet Applications. ActiveX controls suffer in popularity because they are only supported in Internet Explorer. In addition, ActiveX controls are not executed in sandbox. Therefore, they are potential targets for computer viruses and malware making them high security risks. ActiveX is essentially an old technology and has been replaced by Microsoft’s `.Net` framework so little future development is expected from this technology.

ActiveX emerged out of a data-centric vision and has a history itself. From a general perspective, ActiveX has manifested from three evolutionary steps: DDE (Dynamic Data Exchange), and two versions of OLE (Object Linking and Embedding. (Garcia *et al*., 2000)


Media Usage Rights Acquisition

1.licenseserver.com

This website wants to run the following add-on: 'DRM ActiveX Network Object' from 'Microsoft Corporation'. If you trust the website and the add-on and want to a

No available licenses found.

Please enter your login information below

Email address:

Password:

Figure 2.18 Sample Active X application (Filebuzz , 2008)
http://www.filebuzz.com/files/activex_bar_code/1.html accessed 13/09/2008

- ***User interface languages***

Instead of HTML/XHTML, new user interface mark-up languages can be used in Rich Internet Applications. An example of this is Mozilla Foundation's XML-based user interface mark-up language XUL. XUL could be used to develop Rich Internet Applications, though it at this time it only runs on Mozilla-based browsers. Rich Internet Applications user interfaces can also become richer through the use of scriptable scalable vector graphics (though not all browsers can render those natively yet) as well as Synchronized Multimedia Integration Language (SMIL).

- ***Other techniques***

Rich Internet Applications could use XForms to enhance their functionality. Using XML and XSLT along with some XHTML, CSS and JavaScript can also be used to generate richer client side UI components like data tables that can be resorted locally on the client without going back to the server.

2.4 Conclusions

In this chapter we introduced the traditional Web technologies such as the Internet, the World Wide Web and HTML. These technologies are classed as Web 1.0. Following this Web 2.0 was explained as the next generation of the Web. A sub component of Web 2.0 is Rich Internet Applications and this chapter examined how Rich Internet Applications emerged and why they were needed to meet Web user expectations. This chapter also described the various tools for developing Rich Internet

Applications such as Adobe Flash and Silverlight. The next chapter will look at accessibility and how it fits into the Web today.

3. Accessibility

3.1 Introduction

In general *accessibility* describes the level to which a given object (e.g. service, environment or device) is accessible to as many people as possible. It is the ability to access this object which determines its accessibility. Accessibility can specifically focus on people with disabilities and their right of access to entities, often through use of assistive technology. Several definitions of accessibility refer directly to access-based individual rights laws and regulations. Products or services designed to meet these regulations are often termed "Easy Access" or "Accessible". (Di Lucca *et al*., 2005)

In the following sections the general trends around accessibility will be discussed. Contrasts will be drawn between accessibility and usability within websites, as well as looking at inclusive and the universal design of websites. This chapter will also examine assistive technologies which help disabled web users interact with websites. Then the different types of disabilities which require the use of assistive technologies will be examined. The chapter closes by reviewing existing laws on disability acts and how these laws relate to web accessibility.

3.2 Usability

Yates (2005) defines *Usability* as to how easy the web site is for everyone to use, and incorporates design layout patterns that may be "learned" by users who then may explore the site and derive value from its contents. Webdale (2003) extends this definition to cover the ability of users to "interact" with a digital platform, suggesting an exchange element which is facilitated through the technological medium. Lastly, Foley (2003) proposes that thinking should progress the notion of an interactive element to consider technology as a "theatre", characterised by an environment which is interpreted and shaped by the personal experience of the viewer. The concept of a computer as an "agent", suggests Foley (2003), is not only important as a link between the web site and the user, but also has influence on the design strategies adopted by web developers. By forcing developers to contextualise web users in terms of their experiences, background and abilities, they are able to focus on developing a web environment which targets their needs and provides enriched interactivity.

Accessibility should not be confused with usability which is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

3.3 Universal Design

Burgstahler (2006) describes that when creating web sites which can be used by the widest spectrum of potential visitors rather than an idealised "average," Webmasters can apply "universal design" principles. This approach requires that they consider the needs of visitors with a wide range of characteristics individuals with disabilities, older persons, people for whom English is a second language, and those using outdated software and hardware. They should routinely think of the broad range of characteristics their site visitors might have and design their resources to be accessible to them. This is the same approach that modern architects take in designing buildings; they build in ramps, elevators, accessible restrooms, and other features to ensure that the facility will be accessible to individuals with a wide range of abilities and disabilities.

Accessibility is strongly related to universal design when the approach involves "direct access." This is about making things accessible to all people (whether they have a disability or not). However, products marketed as having benefited from a Universal Design process are often the same devices customized specifically for use by people with disabilities. An alternative is to provide "indirect access" by having the entity support the use of a person's assistive technology to achieve access (e.g., screen readers).

3.4 Inclusive Design

Chandrashekar (2007) defines Inclusive design as the design of mainstream products and/or services that are accessible to, and usable by, as many people as reasonably possible on a global basis, in a wide variety of situations and to the greatest extent. As Chandrasekhar's definition suggests inclusive design encapsulates all that is also concerned with Universal Design. Inclusive design is a process whereby designers, manufacturers and service providers ensure that their products work for people irrespective of age or ability. (La Ferla, 2006)

In relation to inclusive design within software development Luke (2001) says "*The major obstacles to accessibility are complexities in page layouts, inconsistencies in item labelling, a lack of instructions for task completion and the absence of consistent and clear functions related to items within platforms. To ensure full accessibility of information, developers need to ensure their platforms conform to the current WAI guidelines. Redundant information display is needed to aid those who are learning disabled. In addition, problems with adaptive or assistive technology need to be acknowledged, addressed and tested to lessen the effects of incompatibilities*."

3.5 Web Accessibility

- Tim Berners-Lee has stated that: *"The power of the Web is in its universality. Access by everyone, regardless of disability, is an essential aspect"*. The domain of Web accessibility defines how to make

Web content usable by people with disabilities and other groups. People with some types of disabilities can use Assistive Technology (AT) to interact with content. AT can transform the presentation of content into a format more suitable to the user, and can allow the user to interact in different ways than the author designed. In order to accomplish this, AT must understand the semantics of the content. Semantics are knowledge of roles, states, and properties, as a person would understand them, that apply to elements within the content. For instance, if a paragraph is semantically identified as such, AT can interact with it as a unit separable from the rest of the content, knowing the exact boundaries of that paragraph. A slider or tree control is a more complex example, in which various parts of a widget each have semantics that must be properly identified for the computer to support effective interaction (WAI-RIA, 2008).

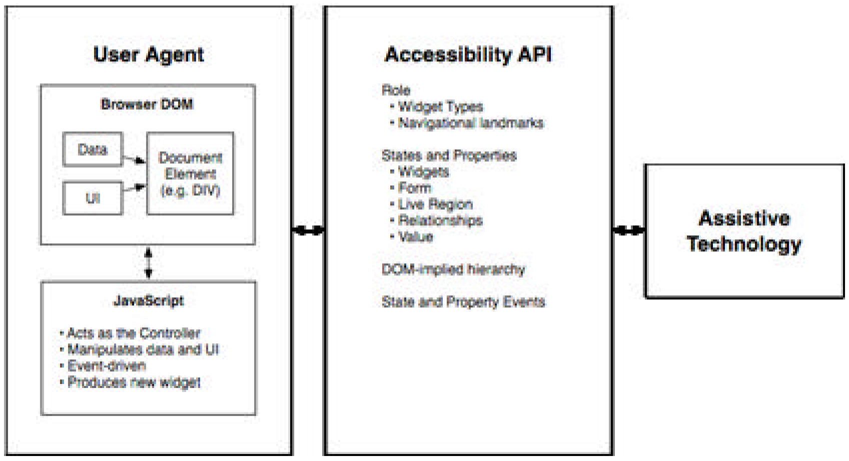


Figure 3.1 ARIA contract model with accessibility APIs (WAI-ARIA , 2008)
WAI-ARIA : Accessible Rich Internet Applications (WAI-ARIA) Version 1.0
http://www.w3.org/TR/wai-aria/ accessed 13/09/2008

Sierkowski (2002) defines Web accessibility as the ability for a person using any user agent (software or hardware that retrieves and renders web content) to understand and fully interact with a website's content. Sierkowski (2002) describes the idea of accessibility as based on more than the implementation of standards; it embodies the idea that everyone has the right to be included in society, regardless of disability, geographical location, language barriers, or any other factor.

Walsh (2006) describes Web accessibility, in its broadest sense as designing a Web page in such as way as to "*ensure all users have access to the information on the page*," including the visually or print impaired. Burgstahler (2006) describes the main web accessibility issues as website visitors who cannot see graphics because of visual impairments; cannot hear audio because of hearing impairments; use slow internet connections and modems or equipment that cannot easily download large file; and have difficulty navigating sites that are poorly organised with unclear directions because they have learning disabilities, speak English as a second language, or are younger than the average user.

The World Wide Web provides an increasing proportion of the services that in many developed societies are necessary for full economic, cultural, and political participation. Recognition of this crucial role has fomented an insistent regulatory climate in many areas of the world that encourages and protects access to the Web by persons with disabilities. The number of persons with disabilities accessing the Web is rising, propelled mainly by the increasing proportion of the aged in populations of developed countries (Fairweather *et al.*, 2002).

Sierkowski (2002) suggests successful delivery of accessibility is dependent on a developer's awareness of the aspects involved. Acknowledging its necessity is vital; becoming familiar with assistive technologies and the tools available to create and verify accessible sites is crucial as well.

3.6 Assistive Technologies (AT)

Assistive technologies (AT) are devices which enable people with disabilities may utilise, to communicate more easily use PCs. A familiarity with the different types of assistive technologies is crucial since thorough testing and verification of accessibility cannot occur without their use (Sierkowski, 2002). Assistive Technologies consists of devices and other solutions that assist people with deficits in physical, mental, or emotional functioning. Assistive Technology devices are items frequently used by people with functional deficits as alternative ways of performing actions, tasks, and activities (Mitchell *et al.*, 1992). Sierkowski (2002) describes three main types of assistive technologies:

- AT For Physical Disabilities
- AT For Hearing Disabilities
- AT For Vision Disabilities

3.6.1 AT for Physical Disabilities

These technologies help users who have impairments related to mobility, motor skills, and their physical being. Alternate input devices - such as foot operated mice, head-mouth pointing devices, sip-and-puff systems controlled by breathing, eye-tracking systems, and unconventionally sized or shaped keyboards—aid users with limited or no fine motor control of their hand (Sierkowski, 2002).

Bigman (2007) explains that blind users who accessed a website using a screen reader had information encoded visually which was inaccessible, functionality requiring the use of the mouse was unavailable, and complex or lengthy content lacking semantic mark-up was inefficient to browse.

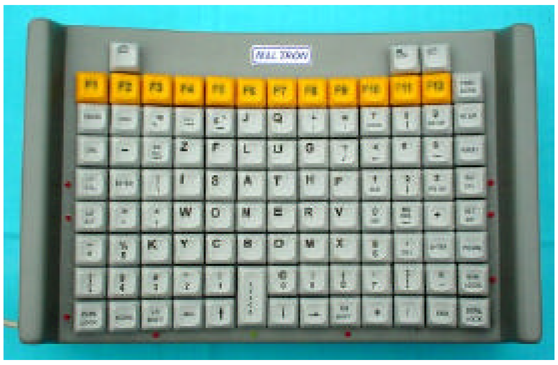


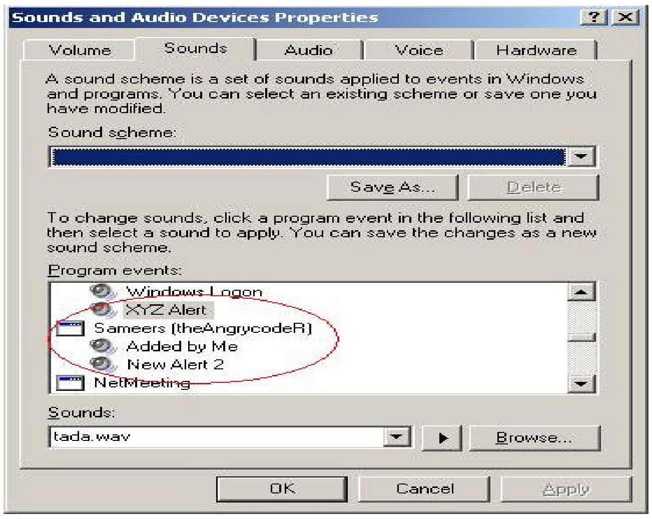
Figure 3.2 The Maltron Single finger or Head/Mouth stick keyboard (Chaffney, 2008)
http://www.chaffey.edu/dps/asst_tech.html accessed 13/09/2008

A typical example of AT is the Maltron Single finger or Head/Mouth stick keyboard which has a unique shape and keyboard layout. The shape matches head movement and the key arrangement minimizes finger or stick activity, raising speed and relieving frustration.

Specialised software is often used in conjunction with alternate input devices. On-screen keyboards, for example, allow users to select keys using a pointing method such as switches or Morse code input systems. Speech recognition software allows input information through speech and predictive dictionaries help by predicting words as the user types them (Sierkowski, 2002).

3.6.2 AT for Hearing Disabilities

Users with hearing disabilities have many options available within existing hardware and software. The major operating systems provide options for visual equivalents of audio alerts; many computer manufacturers provide similar features as well. IBM laptops, for example, have LCD displays which flash a symbol of a speaker whenever a tone is emitted from the computer (Sierkowski, 2002).



Sounds and Audio Devices Properties

Volume | Sounds | Audio | Voice | Hardware

A sound scheme is a set of sounds applied to events in Windows and programs. You can select an existing scheme or save one you have modified.

Sound scheme:

Save As... Delete

To change sounds, click a program event in the following list and then select a sound to apply. You can save the changes as a new sound scheme.

Program events:

Windows Logon
XYZ Alert
Sameers (theAngrycodeR)
Added by Me
New Alert 2
NetMeeting

Sounds:

tada.wav

Browse...

OK Cancel Apply

Figure 3.3 A typical custom alert offe red by Windows OS

3.6.3 AT for Vision Disabilities

Screen readers, such as Freedom Scientifics' JAWS, GW Micro's Window-Eyes and Alva Access Group's OutSpoken, are the most common aids to help users with vision-related disabilities. They capture text as it is written to the display and create an off-screen model (a database of all the text displayed on screen). When a user requests information, it is read from this database rather than from the screen itself (Sierkowski, 2002).

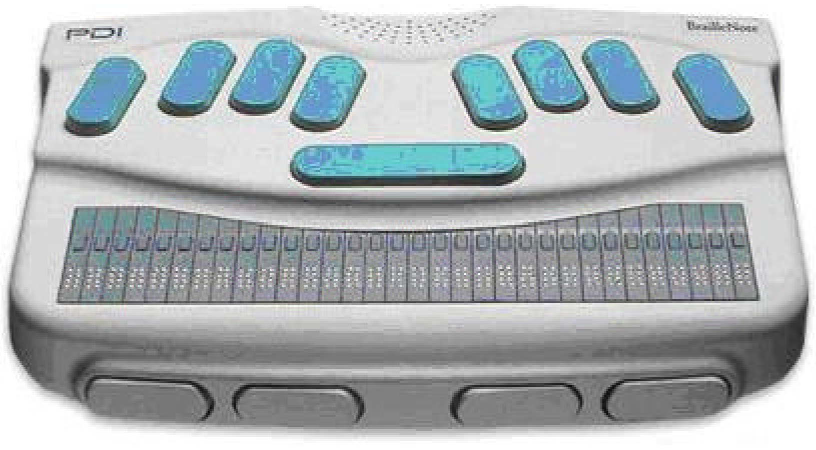


Figure 3.4 A braille re ader (BlinkSOFT , 2004)
BlinkSOFT : BRAILLEX EL Braille Assistant (ELba)
http://www.blinksoft.biz/products/notetakers/elba.html accessed 13/09/2008

Non-visual output devices are used to translate output whether it come from a screen reader or not. Speech synthesizers translate text into audio; Braille display devices use pins to form Braille symbols that refresh and change as the user navigates his or her way. Screen magnifiers and high contrast modifications help those with lesser vision disabilities and are often features built into operating systems or available as inexpensive add-ons (Sierkowski, 2002).

3.7 Legislation

Sierkowski (2002) describes the most important of all reasons why accessibility is important is that it is required by the law. While this researcher doesn't entirely agree with Sierkowski's point, it is important to acknowledge the threat of breaking the law as a motivator to ensure people develop accessible products, is a powerful one.

3.7.1 Irish Disability Act (2005)

The Disability Act 2005 was passed into law with a view to establishing a civil rights approach for people with disabilities. (Gannona *et al*., 2006). Many of the references in the disability act relate to directly to accessible website design within the public service and following extracts are relevant to Rich Internet Applications within the Republic of Ireland:

From Section 27: "*Where a service is provided to a public body, the head of the body shall ensure that the service is accessible to persons with disabilities*".

From Section 28: " *Where a public body communicates in electronic form with one or more persons, the head of the body shall ensure, that as far as practicable, the contents of the communication are accessible to persons with a visual impairment to whom adaptive technology is available*".(Irish Disability Act, 2005)

The Disability Act underlines the level of importance placed on public websites meeting accessibility standards by the Irish government. As Rich Internet Applications become more common, an increasing number of Irish-based websites will fail to meet accessibility laws within Ireland. This gives added importance to improving accessibility semantics of Rich Internet Applications.

3.7.2 UK Disability Discrimination Act (DDA)

The Disability Discrimination Act (DDA), which was enacted in November 1995, and brought to an end a long campaign for the legislation for disability rights. The Act brought in measures to prevent discrimination against disabled people and gave disabled people new rights in the areas of employment, the buying and renting of property and access to goods, facilities and services. (McCaskill *et al*., 2001)

The revised Code of Practice was published in May 2002 and contains the following clarifications and examples:

"*The Disability Discrimination Act makes it unlawful for a service provider to discriminate against a disabled person by refusing to provide any service which it provides to members of the public*."

"*From 1st October 1999 a service provider has to take reasonable steps to change a practice which makes it unreasonably difficult for disabled people to make use of its services*."

"*What services are affected by the Disability Discrimination Act? An airline company provides a flight reservation and booking service to the public on its website. This is a provision of a service and is subject to the act*."

"*For people with visual impairments, the range of auxiliary aids or services which it might be reasonable to provide to ensure that services are accessible might include ... accessible websites*."

To date there have been no cases brought to a UK court, but it is widely believed that the W3C guidelines are an acceptable benchmark for measuring website accessibility. (W3C WAI 2008)

3.7.3 EU Accessibility Legislation

In the EU there are 45 million people with disabilities and the population is ageing (eInclusion 2008). As website activity becomes increasingly central to our lives, disabled people risk severe social exclusion due to a range of technical barriers they face when using the internet. Most of these barriers can be easily avoided if website designers follow a set of simple rules covering site content, structure and coding. Currently the EU is embarking on the i2010 initiative on e-Inclusion, whose goal it is to raise awareness of web accessibility.

European i2010 initiative on e-Inclusion

It is considered that existing EU legislation contains significant hooks for implementation of e-Accessibility. Therefore the 2005 eAccessibility Communication recommends to investigate and increase their current use as a first step and, if it appears that their potential is not sufficient, to consider possible new legislation. In 2003, European Year of People with Disabilities, in the implementation of the new Communication Framework Directive, the "Communication Committee" (COCOM) created the sub-group "Inclusive Communication" (INCOM), to investigate what relevant actions are required. This group continues its work, in particular to analyse the issue in the context of the transposition of the directives at Member States level and the update of these directives later in 2006. (eInclusion Legislation 2008)

eAccessibility

The European Commission is also promoting "eAccessibility" aimed at ensuring people with disabilities and elderly people accessing Information and Communications Technology (ICTs) are on an equal basis with others. This includes removing the barriers encountered when trying to access and use ICT products, services and applications.

3.7.4 Australian Disability Discrimination Act (DDA)

The objectives of the Australian Disability Discrimination Act 1992 are "*to eliminate, as far as possible, discrimination against persons on the ground of disability in the areas of: work, accommodation, education, access to premises, clubs and sport; and the provision of goods, facilities, services and land; and existing laws; and the administration of Commonwealth laws and programs; and to ensure, as far as practicable, that persons with disabilities have the same rights to equality before the law as the rest of the community; and to promote recognition and acceptance within the community of the principle that persons with disabilities have the same fundamental rights as the rest of the community.*" The Act itself does not directly address websites, but case law and supporting documents clearly indicate that websites can be covered under Australian Disability Discrimination Act. It states that equal access to web sites is required by law: Provision of information and other Rich Internet Applications through the Web is a service covered by this Act. This requirement applies to any individual or organisation developing a World Wide Web page in Australia, or placing or maintaining a Web page on an Australian server, and includes pages developed or maintained for purposes relating to employment; education; provision of services including professional services, banking, insurance or financial services, entertainment or recreation, telecommunications services, public transport services, or government services; sale or rental of real estate; sport; activities of voluntary associations; or administration of Commonwealth laws or programs. All these are areas specifically covered by the Act.

In addition to these specific areas, provision of any other information or other services or facilities through the Internet is in itself a service and as such, discrimination in the provision of this service is covered by the Act. It applies to services whether provided for payment or not. Case law and supporting documents clearly state that the W3C guidelines are an acceptable benchmark for measuring website accessibility.

3.7.5 U.S Legal Activities on Web Accessibility

3.7.5.1 The Rehabilitation Act (1973)

Section 508 of the Rehabilitation Act (1973) requires that Federal agencies' electronic and information technology is accessible to people with disabilities. The IT Accessibility & Workforce Division, in the U.S. General Services Administration's Office of Government wide Policy, has been charged with the task of educating Federal employees and building the infrastructure necessary to support Section 508 implementation. Using this web site, Federal employees and the public can access resources for understanding and implementing the requirements of Section 508. (www.section508.gov, 2008)

3.7.5.2 The Americans with Disabilities Act (1990)

The ADA extends the requirements of the Rehabilitation Act to the entire nation, not just programs that receive federal funds. It requires public colleges to ensure communications with persons with disabilities are as effective as communications with others, unless doing so would result in a fundamental alteration to the program or service or in an undue burden; it also mandates a plan of compliance be establis hed and maintained (Sierkowski 2002).

3.8 Guidelines

This document has already made reference to accessibility standards and guidelines. Accessibility guidelines lay down instructions on how to achieve accessible web applications at the time of development. There are many guidelines, produced by numerous bodies, this section will describe the two main bodies whose remit is accessibility.

3.8.1 Accessibility Interoperability Alliance (AIA)

Founded in late 2007, a coalition of leading information and assistive technology companies formed the Accessibility Interoperability Alliance (AIA), an engineering working group dedicated to enabling developers to more easily create accessible software, hardware and Web products. This group aims to reduce barriers to information and communication technologies that currently exist for people with disabilities in today's increasingly digital world.

AIA members will collaborate on specific engineering projects intended to increase interoperability between existing technologies, and will design new technologies or solutions to resolve many of the long-standing challenges associated with developing accessible products. The group aims to yield improved developer guidelines and increase accessibility innovation throughout the industry. The AIA have selected four projects to begin their work

- **Consistent keyboard access**: Developing a set of keyboard shortcuts to provide consistent behaviour to users of assistive technology products in any Web browser

- **Interoperability of Accessibility APIs**: Modifying and/or extending existing accessibility models (Microsoft UI Automation, IAccessible2 and others) to improve the interoperability and exchange of information between IT and assistive technology (AT) products.

- **UI Automation extensions**: Adding features and capabilities to support additional rich document scenarios, address new Web scenarios and more.

- **Accessible RIA Suite (ARIA) mapping through UI Automation**: Designing the mapping of rich Web accessibility information through UI Automation to ensure maximum value for AT products and, therefore, for people with disabilities.

3.8.2 Accessible Rich Internet Applications (ARIA 1.0)

Accessible Rich Internet Applications addresses the accessibility of dynamic Web content for people with disabilities. The roadmap introduces the technologies to map controls, Ajax live regions, and events to accessibility APIs; including custom controls used for Rich Internet Applications. The roadmap also describes new navigation techniques to mark common Web structures as menus, primary content, secondary content, banner information and other types of Web structures. These new technologies can be used to improve the accessibility and usability of Web resources by people with disabilities, without extensive modification to existing libraries of Web resources. (WAI-ARIA Roadmap, 2008)

WAI-ARIA (2008) explains that new technologies can overlook some of the semantics required for accessibility. Furthermore, new authoring practices evolve which override the intended semantics-elements that have one defined semantic meaning in the technology are used with a different semantic meaning intended to be understood by the user.

For example, Rich Internet Applications developers can create a tree control in HTML using CSS and JavaScript even though HTML lacks a semantic element for that. A different element must be used, possibly a list element with display instructions to make it look and behave like a tree control. Assistive technology, however, must re-present the element in a different modality and the display instructions may not be applicable. The AT will present it as a list, which has very different display and interaction from a tree control, and the user may be unsuccessful at understanding and operating the control. (WAI-ARIA, 2008)

The incorporation of Roles for Accessible Rich Internet Applications is a way for an author to provide proper type semantics on custom widgets (elements with repurposed semantics) to make these widgets accessible, usable and interoperable with assistive technologies. This specification identifies the types of widgets and structures that are recognized by accessibility products, by providing ontology of corresponding roles that can be attached to content. This allows elements with a given role to be understood as a particular widget or structural type regardless of any semantic inherited from the implementing technology. Roles are a common property of platform Accessibility APIs which applications use to support assistive technologies. Assistive technology can then use the role information to provide effective presentation and interaction with these elements. (WAI-ARIA, 2008)

Future versions of Rich Internet Applications are expected to allow developers to describe accessible diagrams as well as author defined roles, properties and relations (Thiessen & Chen, 2007)

This role taxonomy currently includes interaction widget (user interface controls) and structural document (content organization) objects. The role taxonomy describes inheritance (widgets that are types of other widgets) and details what states and properties each role supports. When possible, information is provided about mapping of roles to accessibility APIs. (WAI-ARIA, 2008)

Roles are element types and should not change with time or user actions. Changing the role on an element from its initial value will confuse an assistive technology. Platform accessibility APIs, to which the roles are mapped by the browser, do not provide a vehicle to notify the assistive technology of a role type changing. If the old element type is be replaced by a new one, the corresponding element and its sub tree should be removed from the document and a new one inserted containing the new role type. (WAI-ARIA, 2008)

Changeable states and properties of elements are also defined in this specification. States and Properties are used to declare important properties of an element that affect and describe interaction. These properties enable the user agent or operating system to properly handle the element even when these properties are altered dynamically by scripts. For example, alternative input and output technology such as screen readers, speech dictation software and on-screen keyboards must recognize the state of an element (such as: if an object is disabled, checked, focused, collapsed, hidden, etc.).

While it is possible for assistive technologies to access these properties through the Document Object Model [DOM], the preferred mechanism is for the user agent to map the States and Properties to the accessibility API of the operating system. (WAI-ARIA, 2008)

4. Accessibility in Rich Internet Applications

4.1 Introduction

Accessible Rich Internet Applications is an intersection of chapters 2 and 3, discussing Accessible Rich Internet Applications. Published guidelines for achieving Accessible Rich Internet Applications are examined such as the Accessibility Interoperability Alliance and the Web Accessibility Initiative.

4.2 Web 2.0 Accessibility Concerns

Although Web 2.0 applications behave increasingly like desktop applications, they continue to use Web technologies for content transmission, encoding, and presentation. These technologies are used or combined in novel ways that lack the accessibility features that have been built over years into their platform-specific counterparts and are becoming mainstream in “Web 1.0” content. Web 2.0 thus presents significant risks to accessibility (Cooper 2007).

Gibson (2007) takes a less concerned point of view stating that while Web 2.0 can provide enormous benefits, all of the new interaction paradigms are just not immediately accessible. He continues that the basic Web has become fairly accessible to people with disabilities but this was not always the case. Initially people using assistive technologies such as screen readers, screen magnifiers, and alternative input devices had difficulty interacting with the web. Requirements by governments have forced companies to address accessibility and to follow guidelines such as the W3C Web Content Accessibility Guidelines to make the Web accessible. However, new Web 2.0 technologies are pushing the limits of assistive technologies.

Web 2.0 uses scripting and other advanced technologies to create visually appealing, highly interactive Rich Internet Applications. Most of these applications are very visual and rely on mouse interactions to operate. Each Web 2.0 application wants to distinguish itself from others based on a compelling visual design, rich user interface and dynamic interaction. The incremental update of pages which can provide performance and real-time updates are not always accessible to people using assistive technologies. The assistive technology is not always able to interpret the user interaction model, nor is aware of the many updates occurring with a page or how to notify the user of the changes. Even for users able to visually interact with a site, the complicated interactions and updates may be overwhelming or confusing. The use of additional semantics, adaptive interfaces and navigation options can make Web 2.0 more accessible. (Gibson, 2007)

Web 2.0 technologies are not a threat to accessibility, but they do require more informative accessibility semantics. With the proper semantics there is no reason why Rich Internet Applications cannot become fully accessible to all groups.

4.3 Standardisation of New Technologies

Innovation does not threaten standards, it is enabled by standards. One of the core reasons to standardise technologies is to provide a stable platform on which innovation can happen. Various experiments will be tried; some will attain technological or commercial success. Web 2.0 has now achieved a level of success that it makes sense to standardise its technologies, thus improving interoperability and accessibility as well as making it the basis for yet another round of innovations build on top of Web 2.0.

Standardisation has, in fact, already begun with the adoption of `XMLHttpRequest` by the World Wide Web Consortium and the creation of the Accessible Rich Internet Applications specification that allows the mapping of Web 2.0 content to existing, standard accessibility architectures. Another potential activity that could be of benefit to Web 2.0 accessibility would be the creation of a declarative language to provide many of the common interactive features that now must be provided using imperative languages with greater accessibility considerations. As standardisation activities like this proceed, Web 2.0 will become an intrinsically accessible platform and the basis for the next great thing. (Cooper, 2007)

4.4 Semantic Web 2.0 Solutions

The Semantic Web provides a great deal of potential to address issues. Much of the challenge of Web 2.0 is one of semantics. That is, data exists, but is overly embedded in an inaccessible user interface, or is encapsulated in imperative script code that cannot be easily accessed in the absence of a standard API. The Semantic Web standardises formats for the interchange of data. When data in such standard formats is available, third party tools such as assistive technologies can repurpose the presentation much more easily. (Cooper, 2007)

Desktop applications already implement semantics and the accessibility APIs has mechanisms to describe components. The main idea behind a Rich Internet Application is to add the necessary semantic data into the HTML and XHTML mark-up. The browser can then interpret this additional semantic data and provide it to the assistive technology via the accessibility API of the platform. Thus, a screen reader can identify a tree control as such. Each tree item is indicated as well as its hierarchy within the tree and its expanded or collapsed state. In order to make Web 2.0 accessible to all users, more semantic information and behaviours need to be embedded into Web applications and provided to assistive technologies (Gibson, 2007).

4.5 Accessibility in leading Rich Internet Application Development Environments

It is obvious that Macromedia Flash and DHTML content are inaccessible for visually impaired users because of its visual richness. For Flash, a range of efforts have been made to address this issue, such as accessibility guidelines and best practices documents. However, the quantity of accessible Flash content has not been increasing in spite of these efforts (REF). One of the significant issues is the lack of tools to create accessible content. Current Web accessibility technologies are built using XML-based technology infrastructures. In contrast, there is no foundation for investigating inside of Flash content, since it is distributed in a binary format. This characteristic has prevented vendors from developing Flash accessibility technologies. (Saito *et al*., 2006)

The Flash player is installed on over 95% of personal computers worldwide. However, this evolution has been causing severe accessibility issues for people with visual impairments. Rich Internet content is characterised by its visual richness and interactivity. These characteristics make accessibility difficult. Current Web accessibility frameworks are based on HTML and CSS mark-up. Screen readers are reading web pages based on the HTML structure tags and checking tools are evaluating HTML tags as defined in the HTML specifications. Even though Flash content is usually embedded in HTML pages, it is out of the scope of these screen reader frameworks. (Saito *et al*., 2006)

In order to encourage accessibility, the Flash accessibility framework was developed and a lot of effort has been invested in supporting it, resulting in improved Flash authoring tools and the publication of guidelines and documents for best practices. Using this Flash accessibility framework, it is possible to provide an interface similar to that used by GUI applications for Java or Windows applications. This framework covers only relatively simple Flash content, but this level of accessibility enablement is important, especially for compliance with guidelines and regulations. However, the amount of accessible (compliant with the Flash accessibility framework) content has not been increasing. One of the very severe issues is the lack of tools. (Saito *et al*., 2006)

Sato *et al.* (2007) describe Flash and Dynamic HTML (DHTML) as representatives of rich content types, and they appeal to sighted people with their fascinating audio and visual presentations. Although Flash was only a format for simple vector-based animations in 1996, over the following decade, it was enhanced to support high quality audio, scripts with user interactions, built-in GUI components, powerful image processing, and now streaming video support is the latest enhancement.

Screen reader users frequently ask for Flash content to be made accessible. In response to the enactment of Section 508 of the U.S. Rehabilitation Act, an accessibility framework was designed and built into the Flash authoring tools and into the Flash player. This API is used by a wide range of assistive technologies with screen reading capabilities. If an author of Flash content embeds the appropriate accessibility information into the content by using the Flash authoring tools, then the Flash player can expose that information for screen readers through MSAA. This means that this architecture requires content creators to study methods to make content accessible and to apply those methods in their daily authoring tasks. However, it has been very difficult to make the Flash content accessible compared to other formats such as HTML and PDF. The causes can be classified into three kinds of problems. (Sato *et al*., 2007).

First, the Flash content itself is used to create visually intuitive content using animations and movies. It is not static and the content is dynamically created and frequently changed in response to the users' operations and the content's own timeline. Accessibility enabling methodologies for such Rich Internet Applications are not mature. Second, there is no tool to evaluate accessibility for existing Flash content. Currently the only method is to use a screen reader, which imposes a heavy burden on developers. Third, Flash content developers are unfamiliar with accessibility problems. Based on our preliminary investigation, most of them do not even know that Flash has accessibility features.

Due to these problems, not even the basic accessibility enablement requirements have been met in existing Flash content, requirements such as alternative text, controlled reading order, or widgets role information (e.g. for buttons). Unaddressed, Flash accessibility will continue to decrease, and screen reader users will be left out of the richer Internet experiences forever. (Sato *et al*., 2007).

4.6 UI Automation Overview

Microsoft UI Automation is the new accessibility framework for Microsoft Windows, available on all operating systems that support Windows Presentation Foundation (WPF). UI Automation provides programmatic access to most user interface (UI) elements on the desktop, enabling assistive technology products such as screen readers the ability to provide information about the user interface to end users and to manipulate the UI by means other than standard input. UI Automation also allows automated test scripts to interact with the UI. (MSDN, 2008)

Microsoft's implementation of UI Automation is available only on Windows Vista, Windows XP, and Windows Server 2003, since Windows Presentation Foundation (WPF) is only available for these platforms (UI Automation Overview 2008)

On November 7, 2007, Microsoft and Novell Inc., after completion of a year of their interoperability agreement, announced that they would be extending their agreement to include accessibility (Year of

Interoperability Agreement 2008) Specifically, it was announced that Novell would develop an open source adapter allowing the UI Automation framework to work with existing Linux accessibility projects such as the Linux Accessibility Toolkit (ATK), which ships with SUSE Linux Enterprise Desktop, Red Hat Enterprise Linux and Ubuntu Linux. This would eventually make UI Automation cross-platform. On November 7, 2007, Microsoft and Novell Inc., after completion of a year of their interoperability agreement, announced that they would be extending their agreement to include accessibility (Year of Interoperability Agreement 2008). Specifically, it was announced that Novell would develop an open source adapter allowing the UI Automation framework to work with existing Linux accessibility projects such as the Linux Accessibility Toolkit (ATK), which ships with SUSE Linux Enterprise Desktop, Red Hat Enterprise Linux and Ubuntu Linux. This would eventually make UI Automation cross-platform.

4.7 W3C Web Accessibility Initiative (WAI)

The WAI Web Content Accessibility Guidelines (WCAG 1.0) was released in 1999 and encouraged developers to avoid JavaScript when developing accessible Rich Internet Applications. However the WAI now realise the need to provide technologies to map controls, AJAX live regions, and events to accessibility APIs. This has led to a turnaround in the WAI's discouragement of JavaScript. Until now, the W3C/WAI has discouraged the use of JavaScript per the Web Content Accessibility Guidelines. (WAI-ARIA Roadmap, 2008)

Legacy GUI applications have found solutions to the issues of accessibility by using accessibility APIs to interoperate with assistive technologies. Rich Internet Applications however offer no such interfaces. To combat accessibility shortcomings in Rich Internet Applications, the WAI Working Group has released the working draft of ARIAs (Accessible Rich Internet Applications) framework. The working draft includes ARIAs Roadmap, Primer, and Best Practices designed to help developers understand the issues related to accessibility in Rich Internet Applications and how these issues can be solved.

According to the WAI-ARIA Roadmap (2008), to ensure the creation of Accessible Rich Internet Applications the application needs to:

- Allow for discovery of UI components through the use of Semantic Web technologies
- Support today's accessibility architectures
- Be easy to lightweight and scalable
- Allow for the separation of presentation from content
- Allow for passive monitoring of the application by assistive technologies
- Involve both user agent and assistive technology vendors upfront
- Provide developer guidelines as part of the definition process.

- The work done to date has focused on extending current specifications like HTML 4.01 and XHTML 1.x, ensuring the dynamic accessibility of scripted Web content (JavaScript with (X)HTML mark-up).

5. W3C WAI- ARIA 1.0

5.1 Introduction

The Accessible Rich Internet Applications framework (WAI-ARIA, 2008), describes a structure to make dynamic web content and web applications accessible to people with disabilities. It is aimed at dynamic web applications such as AJAX, DHTML and other Rich Internet Applications development technologies. ARIAs aim to define what state information can be controlled by the user and to be able to convey this information to any device. The remainder of this chapter is dedicated to WAI-ARIA 1.0, which has been identified as the key framework for developing accessible Rich Internet Applications.

5.2 Version 1.0

Released 4th February 2008, the Accessible Rich Internet Applications framework 1.0 is the fist public working draft by the Protocols and Formats Working Group (PFWG) and the Web Accessibility Initiative (WAI). Version 1.0 is a single document which combines two previously published the Accessible Rich Internet Applications framework specifications, Roles for Accessible Rich Internet Applications (WAI-ARIA Roles) and the States and Properties Module for Accessible Rich Internet Applications (WAI-ARIA States and Properties). In addition to the Accessible Rich Internet Applications framework, two supporting documents have been released to aid understanding and implementation of accessible RIAs. These are WAI-ARIA Premier and WAI-ARIA Best Practices. The ARIAs Premier describes the reasoning and concepts behind the Accessible Rich Internet Applications framework, while ARIAs Best Practices provides a roadmap for web content developers.

5.3 Key Recommendations

This section will discuss some key recommendations of the Accessible Rich Internet Applications framework. These recommendations have been identified as major contributions to dynamic web content accessibility.

5.3.1 WAI-ARIA Roles

Often in RIA development, the use of anonymous HTML controls such as DIV, SPAN are used to create a standard control, such as a textbox, checkbox or radio button. Web Developers prefer to customize a DIV rather than use the standard HTML element such as a HTML checkbox because using a DIV allows the developer to easily implement dynamic functionality into the DIV control. This makes the job of an AT much more complex as it does not know what type of control the DIV is. The Accessible Rich Internet Applications framework addresses this ambiguity by allowing the developer to add roles to each web control. An example of how a role is defined in a DIV is by adding a new attribute to a DIV, e.g.

```
<div role = "checkbox">.
```

- The roles taxonomy describes the context of a role, e.g., a list item should be inside a list.
- The role taxonomy defines a hierarchy of roles with related properties. For example a directory is a type of list.
- The role taxonomy makes use of OWL to provide a type hierarchy allowing for semantic inheritance similar to a class hierarchy.
- The role taxonomy makes references to related concepts in other specifications.
- The role taxonomy provides descriptions for each role.
- The roles taxonomy describes what states are supported for each role. For example, a checkbox supports being checked.
-

5.3.2 User Interface Properties

These properties define the user interface rendering. There are three properties options which can be applied, these are HIDDEN, SORT and TemplateID

Hidden

This property defines whether or not a property is visible to the user or not. An example would be where a user menu is visible to the user until such time that a specified action (user initiated or otherwise) occurs. This property would be set to 'true' until the action occurs and then the property would change to false to indicate to the AT that the menu is invisible to the user. The framework outlines that Cascading Style Sheets (CSS) may not provide the functionality to change the property value, therefore scripting may be used to change the property value. This captures a dynamic change in opinion by WAI since the initial framework on web accessibility in 2001, they discouraged the user of scripting technologies s such as JavaScript as it hindered the ability of AT to function properly. This was because after a script had modified the property of a control or widget, the AT was not notified of the change. However through the use of appropriate semantics laid out in the WAI-ARIA framework, the WAI have adapted scripting technologies, to update control semantics, which will be conveyed to the AT.

Format

```
[aria-hidden=true] {visibility :hidden;}
```

Sort

The sort property defines if items in a table or grid are sorted in ascending or descending order. Supported values include ascending, descending, none or other where a sort algorithm other than ascending or descending has been applied.

Values:

ascending: items are sorted in ascending order by this column
descending: items are sorted in descending order by this column
none: (default) there is no special sort applied to the column
other: a sort algorithm other than ascending or descending has been applied

Format

```
[aria-sort=ascending]
```

TemplateID

The templateid is a unique identifier, represented as a URI, for which an assistive technology may customize its user interface. Values of this can be set to any appropriate URI.

Used in Roles:	**Section**
	Alert, alertdialog, application, columnheader, Combobox, description, directory, grid, gridcell, Group, img, list, listbox, listitem, log, marquee, menu, menubar, menuitem, menuitemcheckbox, menuitemradio, progressbar, radiogroup, region, row, rowheader, select, status, tablist, tabpanel, timer, toolbar, tooltip, tree, treegrid, treeitem
Value:	Any Unified Resource Identifier (URI)

Figure 5.1. List of controls which can avail of TemplateID

Format

```
[aria-templateid=www.rte.ie/css/]
```

5.3.3 Live Regions

Live regions are web controls on a page which can be updated or refreshed without any user interaction. This poses a particular problem to AT since it needs to know when a region has been updated. Examples of live regions can be a web chat room or Google maps where the content will constantly refresh. The Accessible Rich Internet Applications framework provides properties to warn AT that this control may update itself without the control having focus and it will also provide the AT information on how to handle the control change when it occurs. States and Properties include:

Atomic:	When the region has changed, the atomic property will be used by the AT to decide whether it should present all or some of the changed region to the user.
Busy:	Indicates whether of not a live region is finished updating or not. AT will use this information to make a decision on whether to alert the user of the region change or wait till the change is complete.
Channel:	Used to decide which hardware device should be used to convey the live region update to the user. If an AT user is using a speech synthesizer and Braille then this property will be used to map the change alert to the particular piece of hard ware.
Live:	Describes the types of updates which the user can expect from the region. • *Off* describes the live region as being disabled. • *Polite* would be a background change it would not be necessary to alert the user until they are idle. • *Assertive* would have a more high priority but would not be necessary to interrupt the user. • *Rude* is the highest level and would demand instant user notification. The Accessible Rich Internet Applications framework outlines that this level of interruption could leave the users disoriented and should be implemented with caution.

Figure 5.2. States and properties of Live Regions

5.4 Building Accessible Applications with WAI-ARIA

The Accessible Rich Internet Applications framework describes an application as accessible when:

- Each element or widget has full and correct semantics that fully describes its behaviour (using element names or roles).
- The relationships between elements and groups are known

- States, properties, and container relationships are valid for each element's behaviour and are accessible via the Document Object Model [DOM] and the platform accessibility API.
- There is an element having the correct input focus.

The Accessible Rich Internet Applications framework provides a structure to make the web controls in a RIA semantically rich. User agents will then use the roles and properties semantics to know how to handle each control. Assistive technologies can use the roles and properties to anticipate the behaviour of the elements inside the RIAs; this will help the assistive technology to convey important and sensible information to the user. The assistive semantics will notify the assistive technology of important state changes within the RIA controls.

5.5 Issues with the WAI-ARIA

The WAI-ARIA mark-up for live regions does still have a few issues to be worked out. These include difficulties with determining causality, giving developers the ability to group updates, handling interim updates, and providing higher-level abstractions for web developers. (Thiessen & Chen, 2007). Sometimes, web developers may have an application that needs to update several pieces of information at the same time. If these updates are expected to take a noticeable amount of time, web developers will need a way to tell the AT when the updates are completed and ready to be spoken. By grouping these updates together, web developers can prevent users from hearing the same information multiple times, as well as making a large update more meaningful by having all of its parts presented together. The current WAI-ARIA specification does not provide web developers with this ability. (Thiessen & Chen, 2007)

In most cases, the AT should not announce something that is not currently displayed on the page since, in general, if it is not displayed anymore, then it is not current – skipping it will help prevent users from falling behind. However, there are cases where obsolete items should still be announced. For example, if an Ajax application provided a play-by-play description of a game in the form of a log that only contained the last 5 plays, then all the updates should be read, even if the AT were to fall behind in trying to read all of the updates and a play disappeared from the five current plays on the page before it could be read. It is important in this case to not skip updates that have since disappeared because they contain important information that the user needs to hear in order to make sense of the most current information. There is currently no way [using the Accessible Rich Internet Applications framework 1.0] for web developers to specify whether or not interim changes are relevant. (Thiessen & Chen, 2007)

Finally, the WAI-ARIA 1.0 specification does not have defaults for the live region properties for the roles that it has defined. Having defaults is important as this would give web developers a higher level of abstraction. Rather than trying to manually specify the entire live region properties for each individual widget, web developers should be able to specify what type of widget they have and expect that there be reasonable defaults for how that widget will behave (Thiessen & Chen, 2007). This point is incorrect as the use of defaults will lead to the inaccurate semantics, it is better to have no defaults as it forces the developer to create more sensible semantics.

5.6 Conclusion

Regardless of some issues with the framework, the recommendations and contributions of the Accessible Rich Internet Applications framework are significant. As this research has already found, there are no standard development methodologies in place for creating RIAs. The Accessible Rich Internet Applications framework defines a sensible standard, which can be used to make accessible RIAs at little cost or effort to the developer. It is my view that the WAI-ARIA specification is not perfect as it is still only a working draft. However, I believe the WAI-ARIA will address the issues in time and implementations will be found in version 2.0 of WAI-ARIA framework. This research concludes that just as W3C addressed desktop accessibility issues in the past, the WAI-ARIA will achieve universal success when adopted by the community at large.

6. Research Methods

6.1 Introduction

This chapter will outline the quantitative research experiment used to asses accessibility in Rich Internet Applications. The experiment will have two separate parts, which are built on two leading Rich Internet Applications technologies, Microsoft Silverlight and Macromedia Flash. The key recommendations of the WAI-ARIA will be introduced to both applications. The pilot will attempt to design Rich Internet Applications with key WAI-ARIA features on board. The research will then be able to answer questions relative to accessibility in Rich Internet Applications.

6.2 Approaching WAI-ARIA 1.0

In the previous chapter, key recommendations of the WAI –ARIA were outlined. This experiment will pilot these recommendations in a Rich Internet Applications to gauge the level of difficulty which surrounds introducing accessibility at the design stage of Rich Internet Applications. The pilot software will be developed using two Rich Internet Applications development environments. Sampling accessibility on two separate Rich Internet Applications technologies will validity the empirical observation of accessibility in Rich Internet Applications.

It is clear that Rich Internet Applications do not convey useful information which can be used by multiple assistive technologies. While some Rich Internet Applications development environments do provide limited APIs for selective assistive technologies, this does not solve the problem as it only makes the application accessible to selective assistive technologies. A better solution is for the Rich Internet Applications to expose declarative information which all assistive technologies can use to build a useful narrative description to the end user. This can be achieved though an intermediately handler.

WAI-ARIA 1.0 has recommended features which assistive technologies can benefit from, such as states and properties of objects which define if the control is visible or invisible on the screen. The Accessible Rich Internet Applications framework at a conceptual level is about painting a picture of what the individual Rich Internet Applications is and how the user can drive this application using a universal language, which can be understood by all technologies which wish to interact with the Rich Internet Applications.

6.3 Microsoft Silverlight

Silverlight 2.0 beta has just been released, providing a new development environment for this research pilot to be executed on. Silverlight allows Rich Internet Applications to be developed using XAML and C#, providing a powerful programming environment for developing cutting edge Rich Internet Applications.

The next release of Silverlight 2.0 will have full support for UI Automation which will allow assistive technologies to communicate with a Silverlight Rich Internet Applications through a Microsoft UIA API. Currently however the developer must design a Silverlight Rich Internet Applications with custom accessibility helpers. It must also be noted that Silverlight is one of the newest Rich Internet Applications technologies (released 2007), and accessibility was not an initial aim in the first release. The drive to include UIA in Silverlight 2.0 shows that Microsoft is answering the call of accessibility advocates.

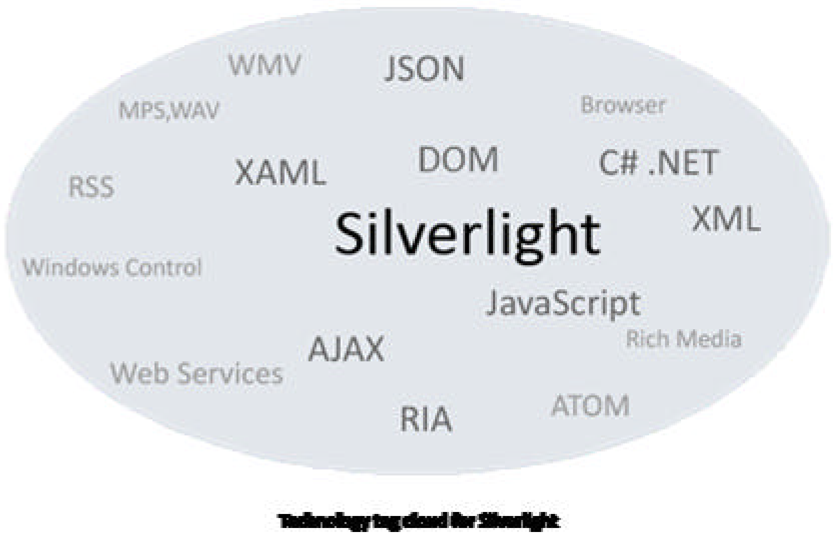


Figure 6.1 A Tag Cloud representing the Silverlight (Rajesh Lal, 2008)
Rajesh Lal: Silverlight
www.irajesh.com/Blogs/

But customised accessibility APIs will not flatten the obstacles which stand in the way of accessible Rich Internet Applications. To prove this point UIA will be used as an example. If an AT is to communicate with a Silverlight Rich Internet Applications, the manufacturers suggest UIA to be used as the intermediately. This is wrong to suggest that all ATs should be compliant with Microsoft's UIA and it very unlikely that AT vendors would become compliant with any software which was designed for the financial benefit of an organisation. A more realistic solution is to create a free to use universal intermediate language.

The Accessible Rich Internet Applications Framework recommends that the visibility status of web controls should be communicated to end users. The Accessible Rich Internet Applications Framework also suggests that when using live region, descriptive messages can be communicated to the end user to keep the user up to date on the status of the region. In terms of recommendations for accessibility in by Microsoft, there is no existing documentation on how to add these accessibility features into a Silverlight application.

6.4 Macromedia Flash

Flash is the leading RIA technology and is currently on version 9. Where a research experiment on RIAs is to be conducted, Flash is definitely going to feature in some part of the experiment because it is by far the most common RIAs technology. Flash content can be found on virtually all leading websites and it is by no means accessible content which can be found. In fact very little accessible Flash applications are available. After looking around the most popular websites I did not find any Flash applications which were flagged as being accessible.

However Macromedia, have an entire section of their website dedicated to Flash accessibility. The documentation found in this site places the responsibility of making accessible Flash applications on the application developer and not on the technology vendor. From best practices to offerings of voluntary product accessibility templates, Macromedia do recognise the importance of accessibility and provide the tools to make accessible Flash content for specific ATs. This approach limits accessible Flash content to a single screen reader (JAWS), there are however many other ATs such as Braille readers, which will not be able to communicate with Flash applications as accessibility is only supported by specific ATs.

A better solution, which would be similar to the proposed experiment with Silverlight would be to create an intermediately language. This language could convey useful navigation information to an AT about what control has focus, helping the disabled user to make a better decision.

The research experiment will attempt create a more accessible Flash application. This experiment will help understand why so little existing Flash content is flagged as accessible. It is clear that Macromedia place the responsibility of accessibility on Flash developers, but are developers finding it too difficult to implement accessible Flash? Perhaps Flash developers are not well tutored in the importance of accessibility?

6.5 Developing a Pilot

Having selected two Rich Internet Applications technologies to pilot the experiment, the next step is to define what accessibility measures need to be built into these Rich Internet Applications. This pilot will attempt to introduce key recommendations of WAI-ARIA to measure the difficulty which is involved in implementing accessible RIAs. WAI-ARIA 1.0 is a realistic benchmark for measuring accessibility, however mirroring the key recommendations within WAI-ARIA are possible with neither Silverlight nor Flash. This is because Silverlight does not render XHTML, but instead runs on a client’s web browser using a dynamic link library, which executes XAML. This approach creates difficulty if we are to introduce the accessible Rich Internet Applications Framework states and properties into the mark up.

Instead, the pilot will attempt to develop the Rich Internet Applications and convey useful accessibility data to the AT via an intermediate XML mark up. By studying the WAI-ARIA key recommendations, the pilot will decide what information is beneficial to the AT and what is not.

7. Implementation of Prototype Applications

7.1 Introduction

To reinforce the scientific finding that accessible RIAs can be developed, the software development requires two applications which are both accessible. One Rich Internet Application will be developed using Microsoft Silverlight 2.0 and the other with Macromedia Flash 9. Both applications are independent Rich Internet Applications. This chapter documents the implementation of both prototype systems and how the accessibility features were added to each.

7.2 Design of Pilot

As discussed in chapter 2, Rich Internet Applications do not have well defined software development methodologies. But research however has proved that a combination of software development and web development methodologies can provide a stable framework for developing Rich Internet Applications. Rich Internet Applications are essentially a richer form of a traditional web application and the chosen software development methodologies used were closely linked to web development.

7.2.1 Rich Internet Applications Software Methodology

As this thesis has already outlined, there are many techniques available for developing Rich Internet Applications and in many cases a combination of technologies are used to satisfy system requirements. This underlines the need for highly structured development methodologies to design and implement reliable Rich Internet Applications. With so many technologies available and some Rich Internet Applications containing a multiple of technologies to achieve the design requirements, development methodologies are vital in building reliable and robust Rich Internet Applications. However, software development methodologies for Rich Internet Applications are not well defined and no development methodologies have been designed specifically for Rich Internet Applications.

Bozzon *et al* (2006) highlights the lack of development methodologies and CASE tools catering for Rich Internet Applications specifically, not withstanding the number of existing methodologies and tools for Web and Hypermedia development, which allow one to specify the application at an abstract level and derive the implementation code (semi-) automatically.

Preciado *et al* (2005) presented the main characteristics of Rich Internet Applications and reviewed several methodologies for the Web, Multimedia and Hypermedia development They concluded that none of the methodologies in these fields are suited to model requirements of the Rich Internet

Applications technology. However, it has been proved that their combination of methodologies presents most of the required characteristics.

Preciado *et al* 2007 again underlined the view that this new kind of Web applications currently lacks complete methodologies and models which aid its design and development. (Preciado, J.C. *et al*., 2007).

7.2.2 Scrum Methodology

Scrum defines a project management framework in which development activities such as requirements gathering, design, coding take place. Scrum development describes a 30-day iteration period as a Sprint. From this period, the Scrum framework has three components: Pre-Sprint, Sprint, and Post-Sprint. The focal point is the 30-day Sprint, in which working software gets developed.

7.2.2.1 Pre Sprint

Scrum uses a series of "backlogs" that identify product features. The Product Backlog contains a list of all business and technology features envisioned for the product. This is a high level document of “what” will be built.

The Sprint Backlog identifies the work to be accomplished by the development team during a 30-day Sprint and it is a subset of the Release Backlog. Even though the development team for this project consists of a single developer, the project can still benefit hugely from scrum processes as it will help define the project. Sprint Backlog is a multi purpose document, at one level it identifies the features, while at another level, it defines the tasks required to implement those features. The Sprint Backlog is developed in a planning meeting. In the meeting, as decision is made on which features are required for the next Sprint. At this point it is necessary to figure out the tasks required to deliver those features.
A task list is documented and then estimates can be made on how much can be accomplished during the Sprint. We then need to ensure that the planned features fit in with the available resources for the Sprint.

The final step in the planning process is to develop a Sprint Goal, which is a business purpose for the Sprint. Essentially a Sprint Goal constantly reminds the developer what the detail tasks should achieve. Without this goal, the team can become overly focused on tasks and lose track of why the tasks are being performed. In addition, keeping the goal in mind encourages the team to adapt to conditions that may arise over the course of the Sprint.

7.2.2.2 Sprint

The principles of a 30-day Sprint are simple: tasks are assigned, hard work is required to accomplish the Sprint Goal, and daily Scrum meetings are held to manage the Sprint. Existing experience and talents are used to deliver results and there are no elaborate plans during the Sprint. Tasks are “locked” and cannot be reassigned during the Sprint (except when disaster arises), this gives each scrum member a clear focus of their tasks and also of their importance. At the end of a Sprint, the Product Owner could choose to throw away all the developed features or reorient the project, but within a Sprint, priorities remain constant. This is known as the "control" part of "controlled chaos."

The reason this is so critical is because in an environment of constant change, there has to be a point of stability. Everything can't change all the time. There needs to be a zone of stability so that the members of the development team can feel that they can accomplish something. Hence, other than in exceptional situations, no changes to the Sprint Backlog are allowed during the 30 days. A key function of the "Scrum Master" (who could be a consultant, coach, or project manager) involves fending off changes during a Sprint to protect the team from getting sidetracked.

Daily Scrum meeting can reenergize a Sprint. Because ad hoc requirements are defined by their uncertainty and change, it is important that checks be exercised daily. The daily Scrum meeting enables the team to monitor status, focus on the work to be done, and raise problems and issues.

Sprint uses daily software builds to raise the visibility of development work and ensure that code modules integrate. If daily builds are good for code, then daily "builds" should be even better for people.

7.2.2.3 Post-Sprint Meeting

At the end of a Sprint iteration, a Post-Sprint meeting is held to review progress, software features can be demonstrated to the customers at this point and a review the project from a technical perspective is held. At the conclusion of this meeting, the Scrum process is repeated starting with a planning meeting for the next Sprint.

7.2.3 Unified Process

The unified process aims to build a robust system architecture incrementally. The process is broke up into iterations so that the "small problems" are easier to manage. Each mini project is an iteration and the key to UP is that each iteration contains all the elements of a normal software development project:

- Planning
- Analysis and design
- Construction
- Integration and test
- An internal or external release

Each iteration generates a baseline that comprises a partially complete version of the final system and any associated project documentation. The baselines then build on each other over successive iterations until the final finished system is achieved.

There are five iteration workflows requirements, analysis, design, implementation and test.

The project life cycle is divided into four phases – inception, Elaboration, Construction and Transition. Within each phase there can be one or more iterations and in each iteration we execute the five core workflows.

7.2.3.1 Inception

Inception is about initiating the project. The primary emphasis in Inception is on requirements and analysis workflows. However, some design and implementation can also be done if it is decided to build a technical, or proof of concept, prototype.

7.2.3.2 Elaboration

Elaboration is about creating a partial but working version of the system. Requirements are refined here and we establish what to build. Testing is conducted on the architectural baseline. The main focus is on requirements analysis and design workflows.

7.2.3.3 Construction

Construction evolves the executable architectural baseline into a complete, working system. The emphasis in this phase is on the implementation workflow. Testing also becomes important at this phase as each new increment builds on the last, both unit and integration tests are now needed.

7.2.3.4 Transition

Transition is about deploying the completed system into the user community. The emphasis is on the implementation and test workflows. Sufficient design is done to correct any design errors found in beta testing. If the project has gone according to plan, this stage will have very little work in the requirements and analysis workflows.

7.3 Accessibility Intermediately XML (AIM)

Design patterns in XML suggest the best practices for the creation of easy to process, readable XML schema. The XML output from the Rich Internet Applications accessibility helper should be readable to a novice XML developer. It is intended that the XML output can be transformed into multiple formats, where each target format may be processed by at least one AT device. After researching the XML best practices, the XML was designed with the following best practice guidelines defined by W3C:

- Do use element declarations, attribute groups, model groups, and simple types.
- Do use XML namespaces as much as possible. Learn the correct way to use them.
- Do not try to be a master of XML Schema. It would take months.
- Do use complex types and attribute declarations.
- Do not use notations
- Do use local declarations.
- Do carefully use substitution groups.
- Do carefully use a schema without the target Namespace attribute (aka chameleon schema.)

The key is to avoid complexity as XML should be simple in structure with respect to the above best practices. The XML encoding used was ISO-8859-15. This standard character encoding is defined by the International Organisation for Standardisation and is the latest standard in the ISO series.

```
<?xml version="1.0" encoding="iso-8859-15"?>
```

Figure 7.1. XML encoding

W3C XML best practices advise the use of namespaces where possible. Namespaces help processors recognise relevant elements and avoid ambiguity within XML. All accessibility semantic elements are prefixed with AIM to tag the element as being specific to the accessibility helper semantics. An XML declaration as follows will be added to the top of each AIM file to define the aim namespace:

```
<access xmlns:aim="http://accessability/helper/">
```

Figure 7.2. AIM Namespace

The namespace used AIM is appended to the elements as follows:

```
<aim:type version="2.0 beta">Microsoft Silverlight</aim:type>
```

Figure 7.3. Implementation of AIM Namespace in XML body

The type element records the Rich Internet Applications technology and the runtime version. In this case the technology is Silverlight and the version is 2.0 (beta). This knowledge will help the AT understand what application it is communicating with. By knowing what Rich Internet Applications technology the semantics are coming from, the AT can make conscious decisions in response to specific events, when fired by a particular Rich Internet Applications technology.

Navigation elements can be found inside the `<aim:context>` element. These semantics will convey accurate up to date navigational information to the AT on the users' current location within the application. Through the creation of a few simple elements the Rich Internet Applications can communicate excellent navigational information to the AT. AIM will hold useful navigational information such as:

- The control the user is currently at.
- what depth this control is on the Rich Internet Applications.
- what are the sibling web controls to the control which has context is on.
- What is the parent control?
- Are there any children elements to the current element?
- Does the application allow keyboard tabbing?

The child elements of `aim:context` conveys the navigation information to the AT. `aim:current` has two attributes called level and type. Attribute level details the current level the user is in the Rich Internet Applications and attribute type is the type of control, a drop down list, checkbox etc. The value of `aim:current` is the value of the attribute, in this case it's the Gambler.

```
<aim:current level="2" type="DataRow">The Gambler</aim:current>
```

Figure 7.4. aim:current records the last data row in application which was accessed by user

The remaining elements within `aim:context` build a navigational view of the Rich Internet Applications, using the user's current position and the states and properties within this control and surrounding controls.

```
<aim:parent>Datagrid</aim:parent>
<aim:children>none</aim:children>
<aim:siblings>10</aim:siblings>
```

Figure 7.5. These elements let AT know what relationships the current element has

The parent control type is defined by `aim:parent`. `<aim:children>` defines the child web controls, while the value of `<aim:siblings>` gives the total number of siblings of the current web control.

AIM will prove at a conceptual level that it is possible for Rich Internet Applications to omit semantics in a language which ATs can use to navigate through a Rich Internet Application. This proof-of-concept is only conceptual and it is beyond the scope of this project to create a language which extracts all the necessary information which AT need for successful Rich Internet Application navigation.

It is intended that AIM can be transformed into a more readable format, which is preferred by the particular AT. For example a screen reader would preface spoken English. The benefit of XML is that the data can be transformed into almost any language or collection. To create spoken English from the XML an XSL style sheet will be applied to the XML which will create a readable spoken English sentence such as “The current element is ‘The Gambler’ and the parent element is a data grid type.”

7.3.1 Silverlight Book Club System

Both pieces of software have one only one non-functional requirement, to be accessibly friendly. Since the experiment is to design the pilots in the most accessible manner possible, we do not yet know if we can deliver an accessibility-friendly piece of software in Flash or Silverlight.

The software application being built on Silverlight is a book sale club system. The user will be able to view available books on a data grid. To obtain further information on a book, the user will select a book in the data grid and further details will be displayed to the right side of the grid. A textbox and button will be situated on the bottom of the Rich Internet Application. The user may wish to enter some feedback text here and click the button to submit the text.

7.3.1.1 Silverlight Functional Requirements

The RIA User Interface should contain a web control which displays a data collection of book items.

User Interface will have a panel on right side of data collection which gives more in depth information on the selected book item. More detailed information to be found here will be Title, Author, ISBN, Price and Purchased date if any. Rich Internet Applications will also have a text box control and a button next to it. When the button is pressed any text within the textbox will be captured. When a selection is made or changed in the data grid using a mouse or by tabbed keyboard focus, an AIM file should be created and saved onto the client PC. The AIM file should be in constructed using the predefined Accessibility Intermediate XML syntax. This file must be accessible by an assistive technology. A tree structure should represent a hierarchical collection of book types. When a user selection is made either by mouse or keyboard the application should generate an AIM file which declares the current states and properties within the tree view. An AIM file should be created when user clicks on the button or when keyboard focus in on the button and a key is pressed down. The Book club system should allow multi-modal access. This can be achieved through keyboard and mouse access.

7.3.1.2 Book Club Class Diagram

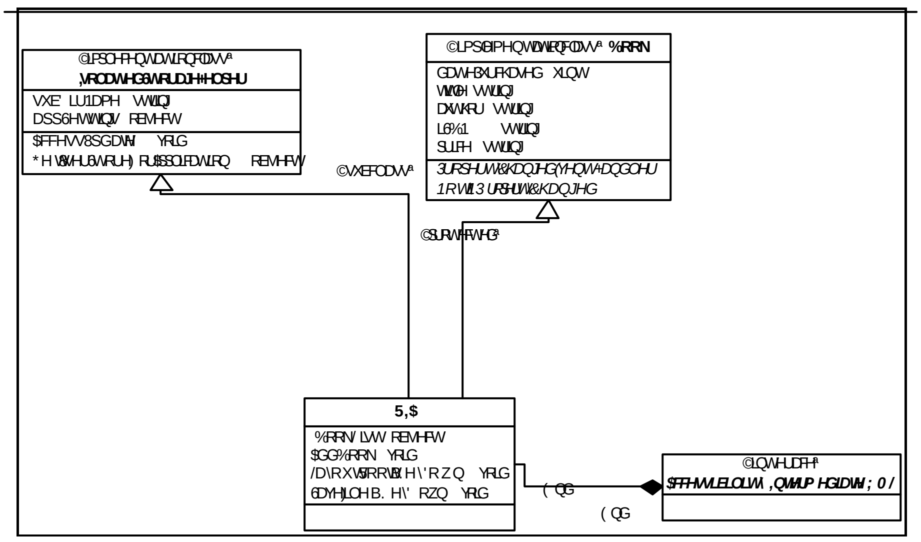


Figure 7.6 shows the class diagram for the Book Club System

7.3.2 Flash Accessibility Demo

Developing an accessible Flash application with basic user interaction features such as adding text items to a text area. This application will have a button in the bottom which will be able to make the buttons bigger in size. Pressing this accessibility button will also make the font size of the text bigger. These features will improve the accessibility for users with visual disabilities and will also be developed using basic action script functions proving that accessible software development can be achieve with relative ease, as long as the developer is aware of accessibility issues.

7.3.2.1 Functional Requirements

Purpose:
To demonstrate the how basic accessibility features can be implemented by a Action Script developer, when the developer is aware of the accessibility issues.

Problem Summary:
Application has a multiple of web controls, which are a standard size. Increasing the size of the controls for a user with visibility disabilities, will make the application accessible for users with visual disabilities.

A recognised button on the application will increase or decrease the size of the controls ,depending on the current size of the controls.

Requirements :

- A Flash 9 application must have a collection of web controls which are varying in size.
- One button, when pressed, will add text into a text area in red font. This functionality will add a new line of text to the text area each time the button is pressed.
- A “make accessible” button, when pressed will increase the size of the important controls and set the visibility of unimportant web controls to false.

User Groups and Roles:
This application should be useable by users without disabilities and users with visual disabilities. Users with visual disabilities will be able to use the application by clicking the recognised “make accessible button” on the bottom of the application.

System Functions:

When the application is run the user for the first time, the user should recognise the “make accessible” button on the bottom left corner of the screen. This mutton with have a large font size making it readable by users with low vision disabilities. Users with low vision disabilities can then click the “make accessible” button and all application controls with increase in size, making them visible.

Development Information:
This application will be developed in Action Script 3, which is the current standard for creating Flash applications. The development will be executed with complete regard to making the application accessible to users with low vision.

7.4 Development Implementation

This section discusses the process of implementing both the Silverlight and Flash applications. All details relating to building both applications can be found in this section.

7.4.1 Introduction

Implementation of two applications required two individual developments. As both applications are built on two individual technologies respectively, the development problem is separated so that the first application development was the Silverlight Book Store and the second development was the Flash Accessibility Demo.

7.4.2 Development Problem

The following sections describe the problems which were encountered at any stage during the development lifecycle. Section 7.4.2.1 deals with problems relating to the Silverlight development and section 7.4.2.2 discusses problems with the Flash development.

7.4.2.1 Book Club System (Silverlight)

The following problems were encountered developing the Silverlight Book Club System:

To prepare for developing on Silverlight 2.0, Silverlight 1.1 alpha was downloaded and used for tutorials. This was because 2.0 beta was not released. However when 2.0 beta went into release, problems were encountered installing the SDK onto my PC. After several failed attempts with no informative errors, a decision was made to uninstall Silverlight 1.1 alpha and only then was a successful Silverlight 2.0 beta 1 installed.

One of the most exiting features of Silverlight 2.0 SDK is the introduction of UI Automation (UIA), which provides an interface for defining states and properties of Silverlight web controls. These states and properties can interpret by any AT which can communicate with UIA API. It was a major goal of this project to implement UIA and evaluate its potential. Prior to the release of Silverlight 2.0 beta 1, research concluded that this feature would be released in Silverlight 2.0., but unfortunately when the Silverlight team released 2.0, they withheld the release of UIA until the next release. This was a big disappointment for this thesis as, UIA attempts to bridge the gap between Act's and Rich Internet Applications.

Rich Internet Applications operate within a sandbox on a client PC, which is a local folder where the application has limited power to within the operating system. This is because Rich Internet Applications are not trusted by the operating system, since they are intruding applications run on the client browser. This limitation presented a problem when implementing a solution, which required the Rich Internet Applications to write a file to the local PC, which could be read by an AT run on the client PC. Microsoft `.Net` library solved this problem by allowing the Rich Internet Applications to implement the Isolated Storage class which can give the Rich Internet Applications access to a storage folder which is considered safe. Using isolated storage enables partially trusted applications to store data in a manner that is controlled by the computer's security policy.

Documentation on Silverlight 2.0 is very difficult to obtain. This is largely attributed to the face that Version 2.0 has just gone into Beta 1 release and it is hoped that more documentation will become available within the coming months.

Silverlight 2.0 beta has limited web controls for showcasing at this time. There is particular unrest within the Silverlight Community about the lack of grids and tree structure controls available on the standard 2.0 toolbox. Research has uncovered that the Silverlight team aim to release a dedicated tree view control before Version 2.0 is put to final release. This presented a problem to this project because the Silverlight pilot designed expected to have a hierarchical tree structure, but this entire tree view would needed to be developed in C# and this development is beyond the scope of this project.

Silverlight uses a new declarative XML based language to initialize structured values and objects. XAML is tricky to implement and a lot of time can be wasted with positioning objects on your canvas. When compared to the ease of use which HTML provides for positioning controls in documents, XAML is a difficult language to implement.

7.4.2.2 Flash Accessibility Demo

The following problems were encountered developing the Flash Accessibility Demo:

Adobe Flash CS3 Professional is a relatively new application and issues were found when finding good quality documentation to get started. While there is a vast amount of online tutorials for using Action Script on the web, very little tutorials are published by Flash making the learning curve steeper than it needs to be.

Changing background colours in the Flash application proved to be a difficult task and required a lot of difficult steps with timelines. Perhaps such a trivial task could be more easily implemented. CS3 is a very powerful development environment, but newcomers to CS3 may find there is a range of complicated processes involved to get a simple Flash application running.

7.4.2 Developing Silverlight Book Club System

The following section discuses the individual components in the application and how they were developed. Creating a Silverlight 2.0 application is a very welcome challenge because Silverlight is a new product from the Microsoft .Net framework. Silverlight 2.0 applications can be created using either Microsoft Expression Blend 2 or Visual Studio 2008. For this pilot Visual Studio 2008 was chosen as the development environment because it allows the developer to easily programme in C# using the code behind files.

7.4.2.1 XAML Implementation

The initial element to be added to the XAML page file was a grid called Layout Root. This is the parent grid and all web controls within the Rich Internet Applications will be placed inside this grid in a hierarchical structure.

Layout Root then defines the rows and columns within the grid. Layout Root has four row definitions. One is the header row, the second is where the data grid will be placed. Third is padding between the data grid and save file panel, while fourth is the save file panel.

Five panels are then defined to complete the Rich Internet Applications layout and the objects can be positioned into the correct positions using row and column indexing. See below for XAML implementation of Layout Root row and column definitions.

```
<Grid.RowDefinitions>
       <RowDefinition Height="15"/>
       <RowDefinition Height="*"/>
       <RowDefinition Height="15" />
       <RowDefinition Height="200" />
 </Grid.RowDefinitions>
 <Grid.ColumnDefinitions>
       <ColumnDefinition Width="15" />
       <ColumnDefinition Width="400"/>             ``
       <ColumnDefinition Width="15" />
       <ColumnDefinition Width="*"/>
       <ColumnDefinition Width="15" />
 </Grid.ColumnDefinitions>
```

Figure 7.7 Grid Definiations in XAML

XAML Data Grid Control

There is a requirement to display a book list with the ISBN number of each title. Silverlight provides a data grid for presentation and manipulation of data collections. Data grids have a vast collection of properties which can be used to customise the control into a book list.

```
<my:DataGrid  x:Name="Books"
```

Figure 7.8 Data Grid definitions in XAML

The book list is required to have two columns and these are defined in XAML as follows:

```
<my:DataGrid.Columns>
            <my:DataGridTextBoxColumn
                    Header="Title"
                    Width="200"
                    DisplayMemberBinding="{Binding Title}"
                    FontSize="14" />
                <my:DataGridTextBoxColumn
                    Header="ISBN"
                    Width="200"
                    DisplayMemberBinding="{Binding ISBN10}"
                    FontSize="14" />
</my:DataGrid.Columns>
```

Figure 7.9 Data Grid Column definitions in XAML

Using the `DisplayMemberBinding` attribute XAML allows a data source field to be bound to a data grid column in the client XAML file. The data source can then be created in the C# code behind file and bound to the data grid in the code behind.

Book Selection Details

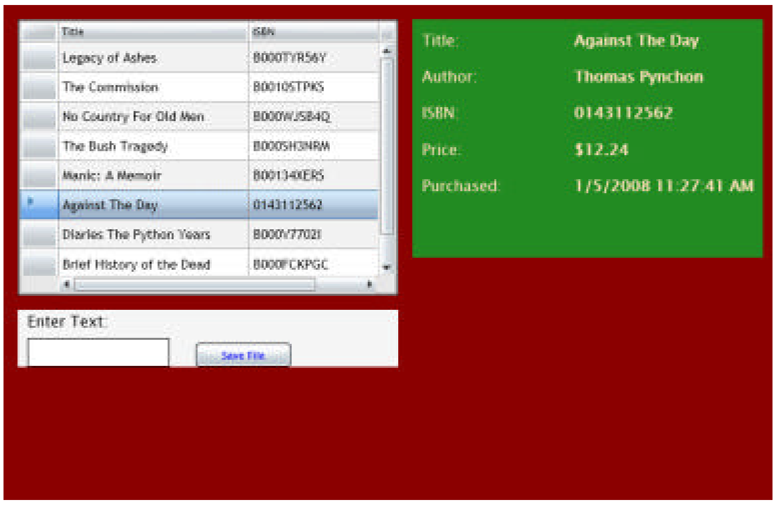
When the user selects a book from the data grid, more details on the user selection will be displayed in the selection details panel to the right of the data grid. A series of text block controls are bound to the selected item of the data source and details of the selected item can then be displayed using the text blocks as follows:

```
<TextBlock Text="{Binding Title}" FontSize="16" Foreground="Wheat"
      HorizontalAlignment="Left"
      FontWeight="Bold"
      Grid.Row="1" Grid.Column="3" />
```

Figure 7.10 Text block definitions in XAML

Save File Panel and Tool Tip button

Three controls are defined, one text block, one text box which allows the user to enter the name of the book and one button which will save the file when pressed.



Title	ISBN
Legacy of Ashes	B000TYR56Y
The Commission	B0010STPKS
No Country For Old Men	B000WJSB4Q
The Bush Tragedy	B000SH3NRM
Manic: A Memoir	B00134XERS
Against The Day	0143112562
Diaries The Python Years	B000V7702I
Brief History of the Dead	B000FCKPGC

Title:	Against The Day
Author:	Thomas Pynchon
ISBN:	0143112562
Price:	$12.24
Purchased:	1/5/2008 11:27:41 AM

Figure 7.11 shows the completed XAML

XAML in built accessibility features – Tool tip and tab index

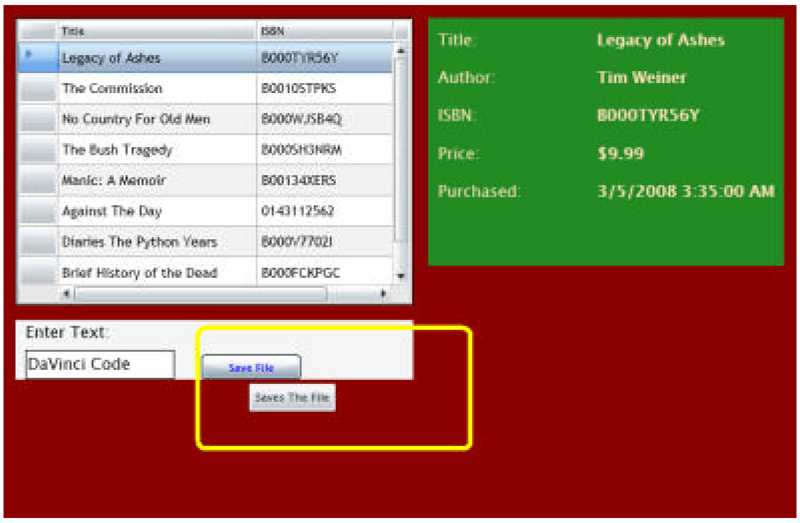
The button uses a standard .Net accessibility feature called tool tip. By using tool tip, when the mouse hovers over the control an informative pop up appears beside the cursor giving more information on the purpose of the button and what action is caused by pressing. The tool tip property must be set by the developer by adding the tool tip property to the control during implementation.

```
   <Button x:Name="SaveFile" Grid.Row="1" Grid.Column="2"
             Margin ="5,5,0,0"  TabIndex="3"
             HorizontalAlignment="Center"
                 Foreground="Blue" Width="100"
                 Content="Save File"
        ClickMode="Release"              KeyDown="SaveFile_KeyDown"
 ToolTip="Saves The File" Padding="30"/>
```

Figure 7.12 Button implementation in XAML

Silverlight 2.0 allows the application to be accessed using the keyboard tabbing. A good XAML feature is the Tab Index property. This property can be set for each control in the application and is a very useful accessibility feature. The value of Tab Index must be numerical to define the order of tabbing within the Rich Internet Application. In the above sample code the Tab Index is set to 3, so the button will be the third control to obtain focus when the user is using the keyboard tab key to navigate the application.

By setting the tab index property of each control it was very easy to make this Rich Internet Applications fully accessible by keyboards only which is of great benefit to disabled users who use ATs such as chorded keyboards.



Title	ISBN
Legacy of Ashes	B000TYR56Y
The Commission	B0010STPKS
No Country For Old Men	B000WJSB4Q
The Bush Tragedy	B000SH3NRM
Manic: A Memoir	B00134XERS
Against The Day	0143112562
Diaries The Python Years	B000V7702I
Brief History of the Dead	B000FCKPGC

Title:	Legacy of Ashes
Author:	Tim Weiner
ISBN:	B000TYR56Y
Price:	$9.99
Purchased:	3/5/2008 3:35:00 AM

Enter Text: DaVinci Code

Figure 7.13 In yellow, the tool tip description “saves the file”, for the button.

7.4.2.2 Business Logic

The implementation of the business logic is in C#. There are a series of files used to realise the functional requirements and these are explained below.

Book Collections

The Book Class is an implementation of a single book object. Each book object has a set of book properties which are set at runtime when a new book object is created. The book class defines the following properties for each book object, title, author, date purchased, ISBN number and price.

IsolatedStorageHelper

A helper class was required to write files to isolated storage. `.Net` contains a library of classes and methods which allow the developer to easily crate an isolated storage directory on the client PC. This library is referenced in the `IsolatedStorageHelper` class.

```
using System.IO.IsolatedStorage;
```

Figure 7.13 Access the isolated storage .Net classes

The first step in isolated storage is to create an isolated storage directory if one does not already exist. The isolated storage constructor handled this logic. This was easily implemented using the C# isolated storage library methods `DirectoryExists()` and `CreateDirectory()`. `DirectoryExists()` checks if isolated storage directory is present on the client PC and returns the Boolean value true if it exists. If the `DirectoryExists()` returns false, we can create a new directory in isolated storage using the `CreateDirectory()` method.

```
    using (var store =
       IsolatedStorageFile.GetUserStoreForApplication())
    {
       if (!store.DirectoryExists(subDirName))
       {
           store.CreateDirectory(subDirName);
       }
    }
```

Figure 7.14 Access the isolated storage .Net classes

Once the isolated storage directory is known, the isolated storage helper can write to this folder is known, the Rich Internet Application can write to this folder. For this a method called `AccessUpdate` was created in the `IsolatedStorageHelper` class

`AccessUpdate` creates a new file stream in the isolated storage directory specified. `CreateFile` method takes two parameters `subDirName` and `fileName`.

`subDirName` is the name of the sub directory to create the file in and `fileName` is the file name to create and write to.

```
   using (IsolatedStorageFileStream newFile =
     store.CreateFile(System.IO.Path.Combine(subDirName,
fileName)))
```

Figure 7.15 Initialise an isolated storage file stream

The stream writer class is then invoked to hold a stream of data. `StreamWriter` class is part of the `.Net` input and output library. StreamWriter takes one parameter, which declares the isolated storage file stream to write to.

```
StreamWriter sw = new StreamWriter(newFile);
sw.WriteLine("<?xml version='1.0' encoding='iso-8859-15'?>");
sw.WriteLine("<Rich Internet
Applications><aim:type>Silverlight</aim:type>");
sw.WriteLine("<aim:context><aim:current level='"+
                  DataGrid.OpacityProperty +"' type='DataRow'>" +
                  selectedbook.Title + "</aim:current>");
sw.Close();
```

Figure 7.16 Initialise a stream writer and write to stream

When the `IsolatedStorageFileStream` is closed, it will write the contents of `StreamWriter` to the isolated storage location defined.

Page.xaml.cs

To create a collection of book objects, `.Net` Generic Collections was used. Generics provide a type safe way of storing well defined types in collections. Generic classes are referenced in the `Page.xaml.cs` file.

```
using System.Collections.Generic;
```

Figure 7.17 Reference a .Net Generic library

Generic collections expose a class called List. Generic List class represents a strongly typed list of objects that can be accessed by index. An empty list of book objects, was declared in the `Page.xaml.cs` file.

```
private List<Book> recentPurchases;
```

Figure 7.18 Initialise generic array of type List

By creating a new instance of the book class and setting the properties of this object, the add method of `recentPurchases` list was used to add the new book object to the generic list. The `AddBook` method was created to take the book object properties as parameter and create a new list item of `recentPurchases`. This method could then be called when needed.

```
private void AddBook(string title, string author, DateTime
datePurchased, string isbn, double price)
{
      Book b = new Book();
      b.Title = title;
      b.Author = author;
      b.DatePurchased = datePurchased;
      b.ISBN10 = isbn;
      b.Price = price;
      recentPurchases.Add(b);
}
```

Figure 7.19 Add new book class

7.4.3 Building Flash Accessibility Demo

The following section discuses the individual components in the application and how they were developed. This application was completely programmed using Action Script 3.0, which is the most up to date version available for development Flash on CS3.

The programming logic was put into a class called `ButtonExample` and class methods were then used as building blocks for the application. The constructor is shown below and calls the three methods shown.

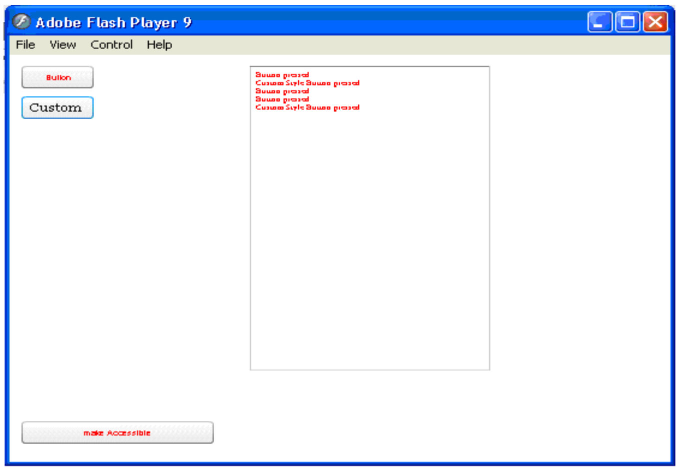
```
public function ButtonExample() {
      createTraceField();
      setupButtons();
      setupfonts();
}
```

Figure 7.20 Initialise Flash controls

`CreateTraceField()` creates a text area to hold the text which gets inserted after a button is pressed. Note the action script method `addChild()` adds the element to the application based on the coordinates specified in `ta.move()`.

```
private function createTraceField():void {
      ta = new TextArea();
      ta.setSize(200, 300);
      ta.move(200, 10);
      addChild(ta);
}
```

Figure 7.21 Set Text Area properties



```
Button pressed
Custom Style Button pressed
Button pressed
Button pressed
Custom Style Button pressed
```

Figure 7.22 Flash Accessibility Demo

`setupButtons()` creates all the buttons and sets a few necessary button properties. This method also adds an event listener to each button. The event listeners will trigger an event when a mouse click is made by the client.

```
b3.addEventListener(MouseEvent.CLICK, buttonClickAccess);
```

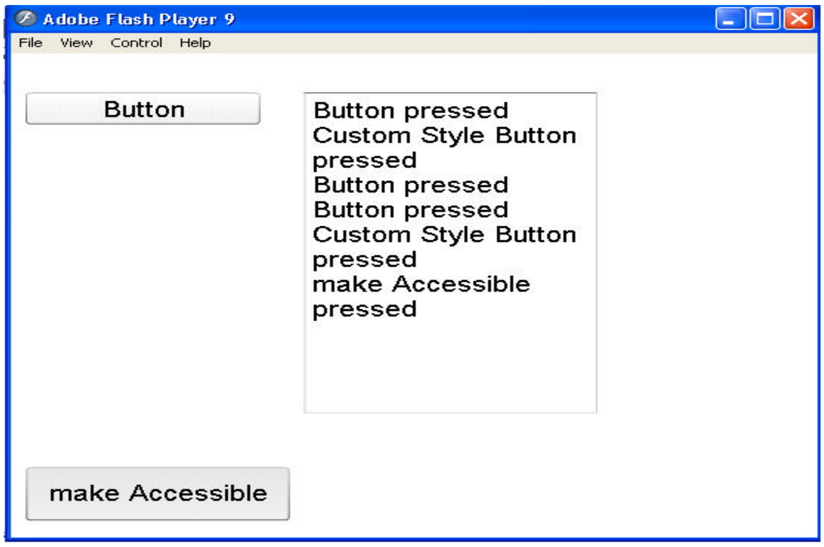
Figure 7.23 Add event listener to button

Below is the `buttonClickAccess` event which will be fired when button b1 is clicked by the mouse. Using event listeners, it is possible to execute business logic when a user action is received by the application. In this case the text area ta is appended some text when the event is fired.

```
private function buttonClickAccess(e:MouseEvent) {
      var button:Button = e.target as Button;
      var tf:TextFormat = new TextFormat();
      tf.color = 0x000000;
      tf.size = 20;
      StyleManager.setStyle("textFormat", tf);
      ta.appendText(button.label + " pressed"+ "\n");

      b1.width = 160;
      b2.width = 260;
      b3.width = 180;
      b3.height = 50;
      b1.height = 30;

      b1.move(10,10);
      b2.move(10,40);

      b2.visible = false;
}
```

Figure 7.24 Event listener will invoke this method

Essentially this event improves the accessiblity of the application when it is fired. `StyleManager.setStyle()` improves the colours being used to display text within the text area. The button sizes are then increased and the visability of button b2 is set to false to make it invisable.



```
Button pressed
Custom Style Button
pressed
Button pressed
Button pressed
Custom Style Button
pressed
make Accessible
pressed
```

Figure 7.25 Application after the make acccessible button is pressed.

7.5 Testing

In this section a series of accessibility tests will be run on both applications. These tests will be able to assess the level of accessibility associated with each application. While the Book Store application which was created using Silverlight technology exhibits more advanced accessibility features than the Flash accessibility demo, at a conceptual level, what is learnt from the accessibility tests in one application can also be applied to other Rich Internet Applications technologies. To validate these tests, several use cases from WAI-ARIA framework are applied as tests cases.

7.5.1 Keyboard Accessibility Support

The following WAI-ARIA use case depicts a scenario when keyboard tabbing can be used to gain accessibility within an Rich Internet Application, *"Alternate input devices are helped when all content is keyboard accessible. The tab index changes achieve this. The new semantics when combined by the Accessible Rich Internet Applications Style Guide work will allow alternate input solutions to facilitate command and control on via an alternate input solution"*(WAI-ARIA 1.0 2008).

Using the Tab Index property for selected web controls in Rich Internet Applications, the developer can create simple keyboard navigation for the user. Tab index also allows the developer to explicitly set the tab ordering within Rich Internet Applications. When the user tabs into a control such as the "save file" button, by pressing the space bar the control event can be fired.

The tabbing functionality within the Silverlight Book Club Store has therefore made the prototype accessible, as it provides an alternative input solution as highlighted by WAI-ARIA 1.0

7.5.2 Low Vision Support

The following WAI-ARIA use case depicts a scenario where users with low vision can be given an alternative input solution by increasing the font size, *"Low vision solutions benefit from ARIA mark-up in that the improved keyboard navigation helps people with extremely low vision. Low vision solutions offer a degree of screen reading functionality (like AI Squared's Zoom text). This is a huge benefit for alternate input solutions as well"* (WAI-ARIA 1.0 2008).

Flash accessibility demo has a button to improve the accessibility of the application in a number of ways. People with low vision may have difficulty reading the text on buttons and within the text area. When the "make accessible" button is activated the size of selected buttons and the text area are increased to help users with low vision. Increasing the size of required web controls has an instant impact for low vision users, who will find the application easier to navigate. This feature was simple to develop and has proven that Rich Internet Applications can be made more accessible to low vision users, if the developer set out to implement low vision help features.

The Flash application default colour is red, which when set against a similar background colour can be difficult to read for colour blind people. The accessible button changes the default font colour to black, set against a white background, making the application more accessible to colour-blind users.

7.5.3 Text to Speech Support

WAI-ARIA recommends *"Rich Internet Applications achieve interoperability with Assistive Technologies"* (WAI-ARIA Roadmap 2008). The purpose of outputting accessibility information into an XML standard is to allow interoperability between Rich Internet Applications and assistive technologies. AIM data is saved to isolated storage on the client's computer, and from there it can be transformed into any format through XSL transformation of AIM. Whether the assistive technology is a Braille reader or a speech narrator, by transformation the XML content of AIM , all ATs can benefit from this XML information.

To create a valid test, an AIM file generated by the Silverlight Book Store application will be transformed with XSL. The transformation output should be in spoken English text which could potentially be read by a speech narrator.

The application was opened in internet explorer and a data grid selection was made. A book is selected from the book list on the application, a AIM file is created by the Rich Internet Application and saved to isolated storage on the client PC. A sample of the AIM file can be seen below:

```
<?xml version="1.0" encoding="iso-8859-15"?>
<aimess xmlns:aim="http://aimessability/helper/">
  <Rich Internet Applications>
    <aim:type version="2.0">Silverlight</aim:type>
    <aim:context>
      <aim:current level="2" type="DataRow">The
Gambler</aim:current>
      <aim:parent>Datagrid</aim:parent>
      <aim:children>none</aim:children>
      <aim:siblings>10</aim:siblings>
    </aim:context>
  </Rich Internet Applications>
</aimess>
```

Figure 7.26 A sample of the AIM file

Using XSL, another output file was then created and saved to a text file. The text file output is shown below:

```
The current user control is a data grid and there are ten
items in this data grid. The data grid has one parent
control. You have selected The Gambler.  There is no child
controls for your selection.
```

Figure 7.27 The text file output from AIM after XSL transformation

This text file was then opened by Microsoft Narrator and the text was read by the narrator. This meant that Rich Internet Applications can output useful information which can be interpreted by an AT, proving that interoperability between ATs and Rich Internet Applications is achievable in Silverlight applications.

7.5.3 Cognitive Disabilities

People with cognitive disabilities exhibit significant delays in measured intelligence, adaptive functioning and academic functioning. These users may find it difficult to make navigational decisions when using a Rich Internet Application which has a lot of web controls, some of which may only be for presentation and could be confusing for users with cognitive disabilities.

This test case asks what can be done to the Flash Accessibility application to make it more accessible to users with cognitive disabilities. One symptom of cognitively disabled people is delays in measured intelligence. A user with this problem could be aided by a more simplistic Rich Internet Application layout. This layout would present the controls to the user in a manner which is orchestrated for easy navigation and displays only the web controls which are necessary.

The accessibility button in the Flash helps this subset of users by removing unnecessary web controls from the Rich Internet Application such as the custom button. This test has proved that if the developer has an understanding for the needs of users with cognitive disabilities, it is very easy to change the appearance of Rich Internet Applications to aid these users. The “make accessible” button also gives the user the option of which layout they wish to access Rich Internet Applications in, normal mode or accessible mode.

8. Evaluation

8.1 Introduction

In this chapter, an evaluation of the key areas will be covered. A research evaluation will outline the key findings related to accessibility in Rich Internet Applications. An evaluation of the software used to create Rich Internet Applications will also explore the accessibility offerings held by both technologies used .

Levels of accessibly within existing Rich Internet Applications technologies can be measured by gauging the ease with which a developer can add accessibility features using the technology. This research evaluation will also discuss the current efforts being made by the EU to draw Web authors into making their websites more accessible to users with disabilities.

8.2 Research Evaluation

The research done in this project was broken into two distinct sections Rich Internet Applications and accessibility. Individually both topics are very board, but when overlapped they define an area which has been ignored by technology vendors called Accessibility in Rich Internet Applications. Thankfully WAI-ARIA seeks to address this problem through a sensible standardisation of semantics within the Rich Internet Applications.

The ambitious document produced by WAI-ARIA tackles accessibility problems in Rich Internet Applications in a well organised way and if it is adopted by the Rich Internet Applications development community at large, it will be successful.

The document advices on web control use and defines a syntax to be integrated within the mark-up of a Rich Internet Application. This research has found that two major Rich Internet Applications technologies do not produce any mark up and can therefore not avail of the declarative syntax recommended by the Accessible Rich Internet Applications Framework. However, the Accessible Rich Internet Applications Framework does provide a very useful best practices document for making accessible Rich Internet Applications and it also advises on how to make individual web controls (such as live regions) accessible.

It was uncovered during this research understanding that there are very few software engineering methodologies which are designed specifically for Rich Internet Applications. This has been a very worrying revelation, as any expectations of more accessible Rich Internet Applications are placed on good software development practices in Rich Internet Application development. But how can accessible Rich Internet Applications be delivered, when there are no development methodologies specifically for Rich Internet Applications. Preciado *et al (*2005) suggested that a satisfactory methodology template can be achieved through a combination of existing software engineering methodologies.

The European Union is currently very active in promoting its eAccessibility programme, which aims at making the web more accessible to people with disabilities. This is welcoming news to web accessibility advocates, but this research has found that the EU recognises the need for new legislation to be introduced to strengthen the laws which apply to web accessibility. There is very little research on accessible Rich Internet Applications, so this is hopefully something that an EU directive could help drawn recognition too. Introducing legislation had a very positive effect on basic web accessibility as it forced companies to create more accessible websites, so this is a lesson the EU should take.

Research has suggested that Web 2.0 community can collectively become more accessible through the standardisation of technologies. This is a very ambitious statement, but could also be very important in flattening out the accessibility obstacles associated with Web 2.0. Research has shown that certain features of Web 2.0 have already been standardised such as the `XMLHttpRequest` by the World Wide Web Consortium and the creation of the Accessible Rich Internet Applications specification. As with most technologies standardisation is a very tricky goal, but it could definitely yield extraordinary benefits if applied to Web 2.0.

It has never been the objective of this author to single out any technology manufacturer for producing very little accessibility features with its development environment, but a lot of research was found which suggests that Flash has not been very proactive in this area. Research has outlined that Rich Internet Applications developers are not well tutored in the area of accessibility development and also that the companies who manufacture Rich Internet Applications development environments have not released any diagnostic tools to asses the accessibility of an application. Research in Microsoft's UI automation has found that, this lack of diagnostic tools in development environments is about to change with Silverlight 2.0 and the UI Spy tool, which can be used to assess the accessibility of Silverlight applications. Today on Adobe CS3, no diagnostics for assessing accessibility in Flash are present.

9. Conclusions and Future Work

Making the web accessibility friendly is a long and difficult journey. The movement is similar in ways to making a building accessibly friendly, where entrance ramps and sufficient door widths for wheelchair access, offer great benefits. However, making buildings accessible is best achieved by adding accessibility features during the construction of the building. The same idea applies for making Rich Internet Applications accessible. Adding accessibility features to a Rich Internet Applications after the application is designed will make the task much more difficult. It would be much better for the developer to design with accessibility in mind.

A major concern is that there are no software engineering methodologies for Rich Internet Applications. Research has found that existing Rich Internet Applications are being designed using no methodology, or in the case where a methodology is used, this methodology is not designed to engineer Rich Internet Applications. This very worrying in terms of accessibility in Rich Internet Applications, because how can we expect developers to design with accessibility in mind, if there are no existing Rich Internet Applications design methodologies?

Rich Internet Applications offer visual experiences which are unparalleled on the Web, so the steady growth of Rich Internet Applications will continue. This growth is reinforced by the inclusion of new Rich Internet Applications technologies by software giants such as Microsoft and Google. Rich Internet Applications development technologies have a big part to play in making Rich Internet Applications accessible. While all Rich Internet Applications technologies have made some efforts to help developers create accessible Rich Internet Applications, the measures taken are often less than satisfactory. By providing applications to help developers test the accessibility of their Rich Internet Applications, these technology vendors can help raise awareness for accessible Rich Internet Applications and also create a more realistic platform for accessible Rich Internet Applications development.

The development of a prototype which created an accessible Rich Internet Applications using two Rich Internet Applications technologies found that accessibility features can be introduced to the application at design time if the developer builds the Rich Internet Applications with accessibility in mind. The components are there to build accessibility features, but testing the accessibility of Rich Internet Applications remains a daunting task, which could be greatly aided by an automatic accessibility test engine.

9.2 Future Work

This section details future recommendations which can aid the accessibility movement of Rich Internet Applications. Section 9.2.1 discusses the potential of AIM as a bridge between AT and RIA. Section 9.2.2 talks about the importance of awareness in accessible software development. The final section discusses the future of WAI-ARIA.

9.2.1 Accessible Mark-up Language (AIM)

Not all Rich Internet Applications can produce mark-up and this fact will prevent WAI-ARIA from achieving widespread success. Since Flash and Silverlight run inside a sandbox, it will be difficult to follow WAI-ARIA tagging recommendations. This is not a criticism of the Accessibility Rich Internet Applications Framework, but more a limitation where a one size fits all solution is not easily achievable.

Accessible Mark up Language (AIM) offers ATs unlimited interoperability with all Rich Internet Applications technologies through an accessible mark up intermediate standard. AIM could definitely adopt the practices and syntax recommended by WAI-ARIA. However it is beyond the scope of this project to build the AIM framework, but through the course of this dissertation it has been clear that limitation of support in WAI-ARIA on Silverlight and Flash ask questions of the adoption of the Accessibility Rich Internet Applications Framework by the Rich Internet Applications community at large.

9.2.2 Accessible RIA Awareness

The testing section in this document has proved that Rich Internet Applications can be made accessible if the developer aims to create an accessible Rich Internet Applications. There is a lack of awareness in the area of accessible web development. A worrying finding has been that developers are not being taught the best practices in making software accessible. Many reasons exist for this shortcoming, one reason is commercial software is developed in a pressurised environment where deadlines are often missed. Another reason is the lack of awareness Rich Internet Applications developers have of accessibility best practices. Perhaps software development students could be taught more about how to create more accessible software. By drawing awareness to these issues, prospective software developers can implement their software with accessibility in mind.

Another method of drawing awareness to web accessibility is through the introduction of new legislation aimed at forcing more accessible software design. With the Web 2.0 movement under way, more and more Rich Internet Applications will replace traditional web applications. This author believes the sooner awareness is drawn to the accessibility issues concerned with Rich Internet Applications, the better.

9.2.3 WAI-ARIA 2.0

The work of the world accessibility initiative to date has seen enormous success throughout the Web community and further success is expected by the release of the Web Content Accessibility Guideline 2.0 (WCAG) to be released in April 2008. WAI-ARIA is a very ambitious document which cannot be implemented in all Rich Internet Applications technologies in the first release. But if the Rich Internet Applications community adopts the principles laid down in this release perhaps bridges can be built in later versions of the Accessibility Rich Internet Applications Framework, which will see Rich Internet Applications giants such as Adobe Flash and Microsoft Silverlight adopting the standards of WAI-ARIA. These two Rich Internet Applications technologies offer powerful development environments so it is expected that the majority of Rich Internet Applications will be produced using one of the two technologies. It is therefore imperative that if WAI-ARIA is to succeed, bridges have to be built which will see the Accessibility Rich Internet Applications Framework being adopted by all Rich Internet Applications technologies and therefore becoming a standard.

Appendix A: Contributors to the WAI-ARIA Framework

Jim Allan (TSBVI)
Chris Blouch (AOL)
Judy Brewer (W3C/MIT)
Charles Chen

Michael Cooper (W3C/MIT)
Donald Evans (AOL)
Aaron Leventhal (IBM)
Alex Li (SAP AG)
Charles McCathieNevile (Opera)

Stefan Schnabel (SAP)
Richard Schwerdtfeger (IBM)
Lisa Seeman (UB Access)

Kentarou Fukuda (IBM)
Alfred S. Gilman
Andres Gonzalez (Adobe)
Georgios GrigoRich Internet Applications (SAP AG)
Jeff Grimes (Oracle)
Jon Gunderson (UIUC)
Dave Pawson (RNIB)
David Poehlman (State of MD)
Gregory Rosmaita (The Linux Foundation)

Janina Sajka (The Linux Foundation)
Ryan Williams (Oracle)
Gottfried Zimmermann (Access Technologies Group)

Bibliography

Saito, S., Hironobu, T., Cieko, A. (2006) “Transforming Flash to XML for Accessibility Evaluations”

Cooper, M. (2007) “Accessibility of Emerging Rich Web Technologies: Web 2.0 and the Semantic Web”

Gibson, B. (2007) “Enabling an Accessible Web 2.0”

Industry Alliance Aims to Increase Accessibility, Interoperability and Innovation http://www.accessinteropalliance.org/newsevents/pr121007.html accessed 27/03/2008

WAI-ARIA Roadmap (2008) http://www.w3.org/TR/WAI-ARIA-roadmap/ accessed 27/03/2008

WAI-ARIA (2008) http://www.w3.org/TR/WAI-ARIA/ accessed 27/03/2008

Thiessen, P., Chen, C. (2007) “Ajax Live Regions: Chat as a Case Example”

MSDN (2008) “UI Automation Overview” http://msdn2.microsoft.com/en-us/library/ms747327.aspx accessed 26/03/2008

Burgstahler, S (2006) “Web accessibility: guidelines for busy administrators”

Yates R (2005) “Web site accessibility and usability: towards more functional sites for all“

Webdale, J. (2003) “Accessibility: New Media Age”

Foley, A. (2003) “Integrating accessible design into the educational web design”

Chandrashekar, S. (2007) “A Model for Inclusive Design of Digital Libraries”

Fairweather, P.G., Hanson, V.L., Detweiler, S.R., Schwerdtfeger, R.S. (2002) “From Assistive Technology to a Web Accessibility Service”

Bigham, J.P. (2007) “Accessmonkey: Enabling and Sharing End User Accessibility Improvements”

Sierkowski, B. (2002) “Achieving Web Accessibility”

McCaskill, K., Goulding, A. (2001) English public library services and the Disability Discrimination Act

UK DDA “Code of practice” http://www.equalityhumanrights.com/Documents/Disability/Services/DRC%20Access%20code%20of%20practice.pdf accessed 26/03/2008

World Wide Web Access, Version 3.2 (2002) “Disability Discrimination Act Advisory Notes”

U.S. Government (2008) “508 Law” http://www.section508.gov/ accessed 26/03/2008

Carughi, G.T., (2007) “Modelling data-intensive Rich Internet Applications with server push support”

Jadoul, R., Plichart, P., Switlik, J. and Latour, T. (2006) “eXULiS – a Rich Internet Application (RIA) framework used for eLearning and eTesting”

Bozzon, A., Comai, S., Fratemali, P., Carughi, G.T. (2006) “Capturing RICH INTERNET APPLICATIONS Concepts in a Web Modeling Language”

Preciado J.C., Linaje, M., Comai S., Sánchez-Figueroa, F. (2007) “Designing Rich Internet Applications with Web Engineering Methodologies”

Preciado J.C., Linaje, M., Comai S. (2005) “Necessity of methodologies to model Rich Internet Applications”

Morales-Chalarro, R. , Linaje, M., Preciado J.C, Sánchez-Figueroa, F. (2008) “MVC Web design patterns and Rich Internet Applications”

Bozzon, A., Comai, S., Fraternali, P. and Carughi, G.T. (2007) “Conceptual Modelling and Code Generation for Rich Internet Applications”

O’Reilly T (2005) “What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software”

W3C XML Schema Design Patterns (2002) “Avoiding Complexity” http://www.xml.com/pub/a/2002/11/20/schemas.html accessed 26/03/2008

Lammarsch, T., Aigner, W., Bertone, A., Miksch, S., Turic, T. (2008) “A Comparison of Programming Platforms for Interactive Visualization in Web Browser Based Applications”

Paulson, L.D. (2005) “Building Rich Web Applications with Ajax”

Meliá, S., Gómez, J., Pérez, S., Díaz, O. (2008) “A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA”

Rinc, S (2008) “Towards standard semantic image annotation and search”

Garcia, R.C. , Heck, B.S. (2000) “Enhancing Engineering Education on the Web: The use of ActiveX Controls and Automation Server Technology”

La Ferla, B. (2006) “Whose design is it anyway?”

Luke, R. (2001) “Courseware Accessibility: Recommendations for Inclusive Design”

Gannona, B., Nolan, B. (2006) “The impact of disability transitions on social inclusion”

Lawton, G (2008) “New Ways to Build Rich Internet Applications”

Velasco, C., Denev, D., Stegemann, S., Mohamad, Y. (2008) “A Web Compliance Engineering framework to support the development of accessible Rich Internet Applications”

Mercier, D., Selina, D. (1998) “What the internet can do for Yo u”

Weaver, A.C. (1997) “The Internet and the World Wide Web”

Korpela, J. (1998) “Lurching Toward Babel: HTML, CSS and XML”

W3C HTML 5 (2008) “W3CA vocabulary and associated APIs for HTML and XHTML”

Mirella M., Zografoula V., Vassilis T. (2005) “Tree-Pattern Queries on a Lightweight XML Processor”

O’Reilly, T (2005) “What Is Web 2.0 Design Patterns and Business Models for the Next Generation of Software”
http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
accessed 23/03/2008

Ketterl, M., Mertens, R., Vornberger, O. (2007) “Vector Graphics for Web Lectures: Comparing Adobe Flash 9 and SVG”

Appcelerator RIA Framework

http://www.ajaxwith.com/Appcelerator-RIA-Framework.html accessed 23/09/2008

Cook, R. (2007) “The Design of a Java Phone Programming Environment”

Iffat, H., Howard, H., Berdenia, S. (2000) “Techniques for Obtaining High Performance in Java Programs”

MS Silverlight (2008) http://silverlight.net/ accessed 20/09/2008

Di Lucca, G.A., Fasolino, A. R., Tramontana, P. (2005) “Web Site Accessibility: Identifying and Fixing Accessibility Problems in Client Page Code”

W3C WAI Policy (2008) http://www.w3.org/WAI/Policy/
Accessed 23/09/2008

eInclusion (2008) http://ec.europa.eu/information_society/policy/accessibility/z-techserv-web/index_en.htm accessed 23/09/2008

eInclusion (2008) “Legislation for eInclusion”
http://ec.europa.eu/information_society/policy/accessibility/policy/legislation/index_en.htm Accessed 23/09/2008

Microsoft, UI Automation Overview, accessed 2007-02-07.
http://msdn.microsoft.com/en-us/library/ms747327.aspx

Microsoft and Novell Celebrate Year of Interoperability, Expand Collaboration Agreement
http://www.microsoft.com/presspass/press/2007/nov07/11-07MSNovell1YearPR.mspx