Technische Universität München
Fakultät für Informatik
Lehrstuhl für Software Engineering betrieblicher Informationssysteme (I 19)
Prof. Dr. F. Matthes

Systementwicklungsprojekt

Evaluation of the RIA Technology Macromedia Flex for User-Centered Social Software

Yi Zheng

Aufgabensteller:	Prof. Dr. Florian Matthes
Betreuer:	Vanda Lehel Sheng Wei
Abgabedatum:	22.05.2006

Table of Contents

1 Introduction

This chapter is an introduction to Rich Internet Application technologies. It also provides information about the aims and organization of this report in the subsequent sections.

Rich Internet Applications (RIAs) combine the ubiquity of web browser-based applications with the user interface of traditional desktop applications as well as the interactive, multimedia communication to favour the enterprise and business application development. Desktop applications offer an interactive user interface for validation and formatting, fast interface response time without page refresh, common user interface behaviours such as drag-and-drop and the ability to work online or offline. Meanwhile advantages of the Web applications include: instant deployment, cross-platform availability, the use of progressive download for retrieving content and data, the magazine-like layout and wide Internet standards adopted. The best of communication means incorporating two-way interactive audio and video [Duhl03]. With RIAs, developers benefit from the simplicity and low-cost of server based deployments while end-users take advantage of a true rich application with dramatically improved performance and an enhanced user experience: more reliable, robust, responsive and effective.

RIAs represent the trend of the future for enterprise application development which is manifest by nearly all technologies from Adobe Flex with Flash Platform, Ajax – related web frameworks, Mozilla's XUL, Microsoft's Windows Presentation Foundation (formerly code named "Avalon") , Sun's Java Web Start.

1.1 Aims of this Report

This report will address the situation of Rich Internet Application Development. Some solutions and technologies are described that have been incorporated during prototyping the application *Social Organizer* based on Adobe Flex framework. Moreover, the *Social Organizer* user interface is presented as an example for analysing how Flex toolsets can provide user enhanced experience and how this can be achieved seamlessly. Finally, it will also be briefly pointed out how Flex 2.0 and other frameworks can be used to improve the features of the *Social Organizer* user interface.

1.2 Organisation of this Report

This report is divided into four chapters:

The second chapter will first educe the term RIA through briefing the disadvantage of current web application, and then mainly explain some major solutions of RIA, finally conclude with a detailed feature comparison of their benefits and limitations respectively.

The third chapter will describe the fundamental concepts of User-Centered Social Software which the prototype *Social Organizer* is based on. After that the focus will be put on the implementation of the *Social Organizer* UI by using Adobe Flex and some main features and mechanisms of Flex.

The last chapter will summarize the subjects involved in this report and the success made by using one of mentioned RIA methods: Flex. Moreover, some suggestions from practice will also be given to help to improve the future development of the application *Social Organizer*.

2 Rich Internet Applications

*This chapter provides the technical background for the UI implementation of the **Social Organizer**. It explains the limitations of traditional web applications, discusses the features of rich Internet applications and the technical requirements. Moreover, it explores the current popular RIA solutions: Java-extending web applications, XML UI, Ajax-related web framework, Avalon, and Adobe Flex framework as well. A comprehensive comparison between these solutions is also given at the end in order to help choose appropriate effective method for building a rich Internet application.*

2.1 Limitations of Traditional Web Applications

Traditional web applications are based on **Client/Server** network architecture with **thin clients**. A client is separated by such “two-tier” architecture from the server. The thin client has little or no application logic and depends primarily on the central server for processing activities. So far this model has been proven successful in aspects of easy manageability, high security, centralized administer controls, low hardware cost etc. However it still suffers from some limitations as:

- Inflexibility

  A route is fixed like this: all interaction with the application must pass through the server and then the server responds to get a page reloaded.

- Bandwidth issue

  Like any network, the available bandwidth for any user depends on the number of total users in the network. By thin client all processing need to be carried out by the server. The increasing user number accompanies with increasing processing commands and response actions result in network traffic.

- Shortage of application rich interfaces and poor multimedia performance

  Graphics intensive applications or multimedia programs which require high levels of processing may not work well in networks based on thin client because processing is performed by the server and is shared with the rest of the network. Recent advances in both processing technology and server technology have improved this to some extent, but there are some problematic areas e.g. real time applications will encounter latency on a thin client network [Bect04].

In opposition to thin clients rich clients need less network bandwidth and fewer server requirements while offering better multimedia performance and possess more flexibility. By using a client side technology the client’s computer can execute more complex instructions. Figure 2.1 compares the functionalities between thin client and rich client. The thin client normally include only presentation engine, thus even the presentation logic has to be handled by server side. The fat client or rich client can process not only the presentation logic but some or whole business logic of an application with ease. Obviously, the load of the server can be effectively reduced.

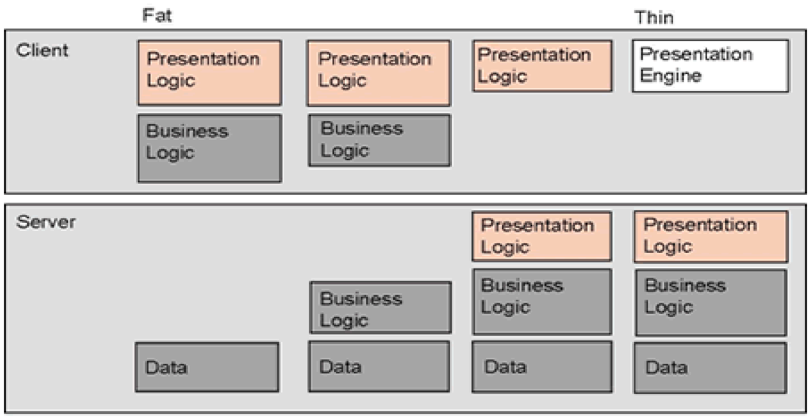


Figure 2.1: Differences of Rich client and thin client [Dome06]

2.2 Term and Idea of RIA

History of RIA

The concept of Rich Internet Application derives from the relevant influence and benefits which web brings to the businesses, education and government worldwide with rapid growth in the past ten years. Even though until now the Internet's potential as a platform for commerce, communications, and business automation has to be constrained by the limitations of today's user experience. The key to improve the Internet's potential concerns two aspects: to deliver end-user more effective experiences through the browser; and to extend the capabilities of the web to provide richer, more interactive, and more responsive user interfaces that can be deployed not just to personal computers but across many portable devices [Macr03]. To meet this requirement, a new generation of internet applications – Rich Internet Applications – is emerging. Macromedia whitepaper of March 2002 [Alla02] introduced this term officially. Several years before that the concept was named like:

- Remote Scripting , by Microsoft, circa 1998
- X Internet, by Forrester Research [Forr06], in October 2000,

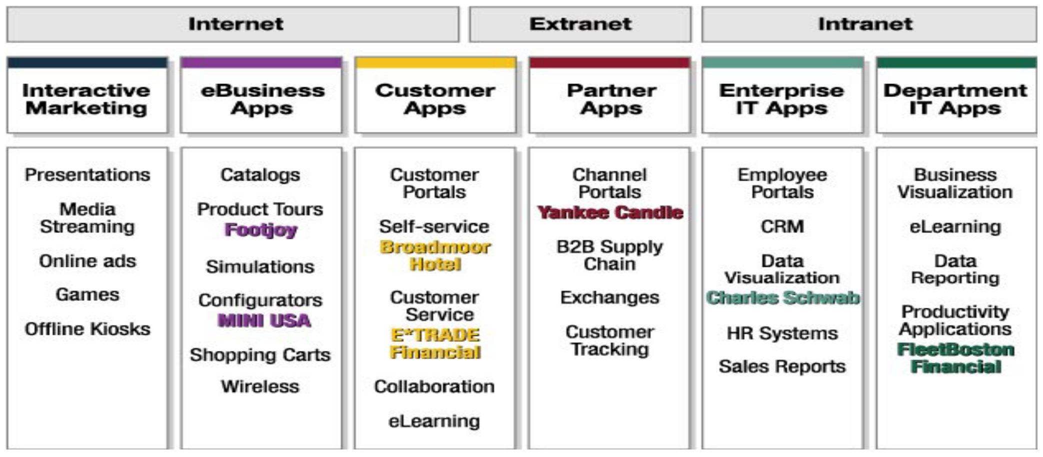
- Webtop, coined by Hummingbird Ltd [Humm06] for their DM WebServer product in February 2002,
- Rich (Web) Clients,
- Rich Web Application

Definition of RIA

Rich Internet Application (RIA) which runs on top of rich client is a combination of web applications and traditional desktop applications; so that it takes advantages of rich client technology to offer more intuitive, responsive and effective user experiences on the web. It combines the media-rich power of the traditional desktop with the deployment flexibility and content-rich nature of web applications. The client is capable of doing more than just rendering pages, such as performing computations, sending and retrieving data in the background asynchronously from the user's requests, redrawing partial screen, using audio and video in a tightly integrated manner [Duhl03], and all those can be done independently from the server or a backend it is connected to. Most RIAs run within a browser and many run within a web page along with HTML content. Still HTML plays a big role in delivering content, user interfaces and navigation [Macr03]. RIAs can also run locally in a secure environment like applets and virtual machines or be "occasionally connected" wandering in and out of hot-spots [Wiki05].

Spectrum of RIA

The spectrum of potential uses for Rich Internet Application (RIA) is fairly broad. Figure 2.2 explores the scope ranging from externally interactive sites, to customer and partner facing applications, to internal enterprise and departmental applications.



Internet			Extranet	Intranet	
Interactive Marketing	eBusiness Apps	Customer Apps	Partner Apps	Enterprise IT Apps	Department IT Apps
Presentations	Catalogs	Customer Portals	Channel Portals Yankee Candle	Employee Portals	Business Visualization
Media Streaming	Product Tours Footjoy	Self-service Broadmoor Hotel	B2B Supply Chain	CRM	eLearning
Online ads	Simulations	Customer Service E*TRADE Financial	Exchanges	Data Visualization Charles Schwab	Data Reporting
Games	Configurators MINI USA	Collaboration	Customer Tracking	HR Systems	Productivity Applications FleetBoston Financial
Offline Kiosks	Shopping Carts	eLearning		Sales Reports	
	Wireless				

Figure 2.2: Spectrum of Rich Internet Applications [Duhl03]

Examples of RIA

Here some examples can help to better understand how Rich Internet Applications advance Internet application development and benefit these areas.

- **Broadmoor Hotel:** The Broadmoor uses a flash-based Rich Internet Application to deliver a better user experience for online reservations. While the original reservation system used five HTML pages, the new system provides a single, intuitive screen. By moving the reservation from HTML to a rich client technology, the Broadmoor reduced the average time for their customers to complete a reservation and increased the number of online reservations (www.broadmoor.com).

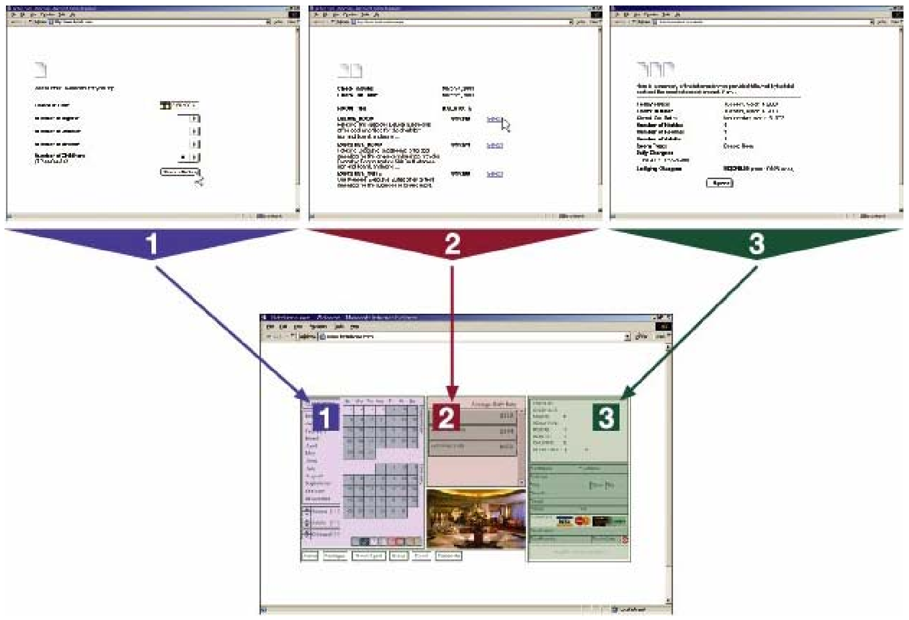


Figure 2.3: Single Screen Interface [Duhl03]

- **Google Map:** Google Map [Gmap05] features a draggable map that can be zoomed in to show detailed street information. It is an example of web applications using Ajax-related approach.

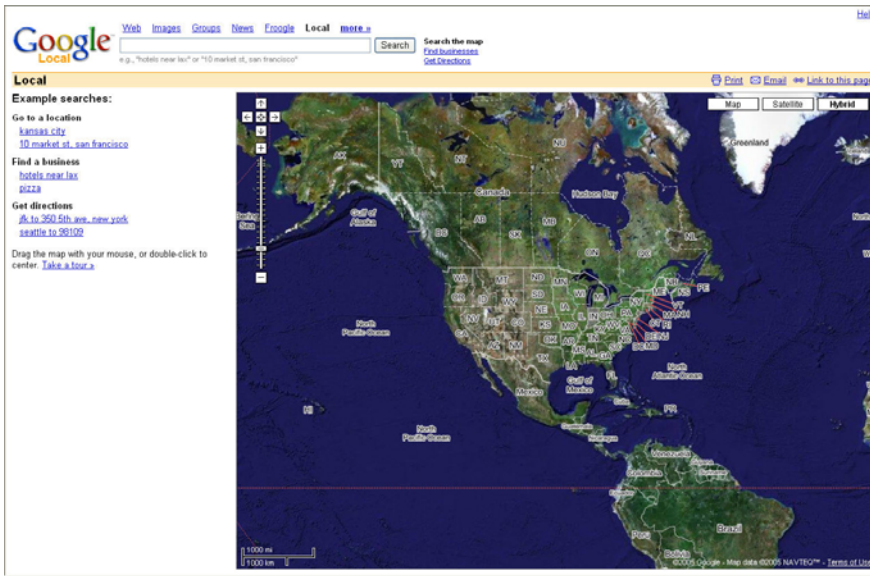

Google Local
Web Images Groups News Froogle Local more »
Search
Search the map
Find businesses
Get Directions
Local
Print Email Link to this page
Example searches:
Go to a location
kansas city
10 market st, san francisco
Find a business
hotels near lax
pizza
Get directions
jfk to 350 5th ave, new york
seattle to 98109
Drag the map with your mouse, or double-click to center. Take a tour »
Map Satellite Hybrid

Figure 2.4: Google Map (Hybrid) [Gmap05]

- **Yahoo maps:** a series of web applications of Yahoo which are based on the Adobe Flex framework. They enable user to find interactive sites that can provide maps of cities and towns around the world, driving directions, road trip plans, and local information. Using those people will feel even easier and more fun to get where they are going.

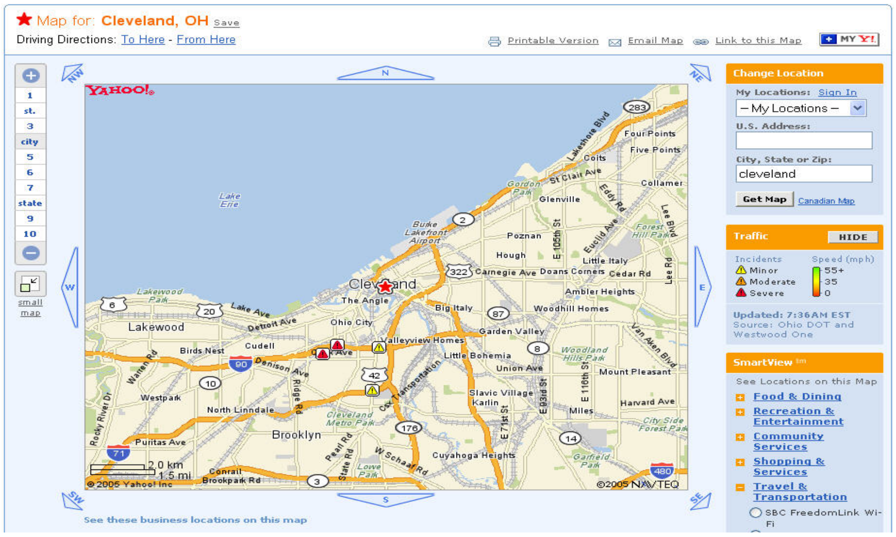
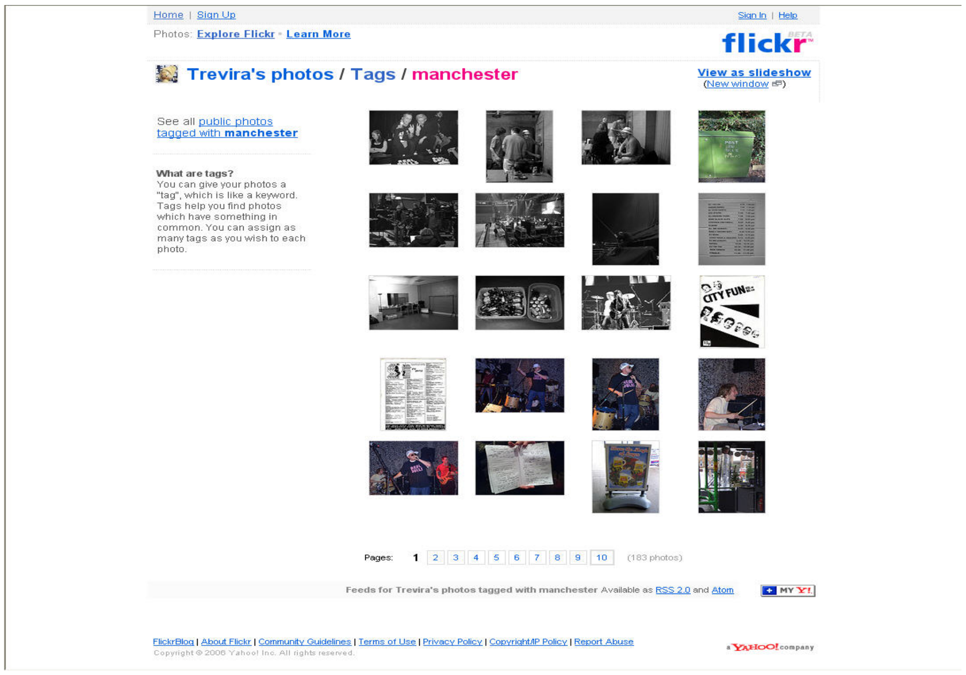


Figure 2.5: Yahoo Map [Ymap05]

- **Flickr:** Using Ajax and Flash technologies this photo-sharing community enables users to upload hundreds of photos and tag each photo with descriptive words. Other user can then search on these tags, enabling them to find and comment on the photos of other users. Flickr's community and addictive sharing features have attracted millions of users. Even better, Flickr exposes a rich set of Web services.


Home | Sign Up
Sign In | Help
Photos: Explore Flickr • Learn More
flickr BETA
Trevira's photos / Tags / manchester
View as slideshow
(New window)
Pages: 1 2 3 4 5 6 7 8 9 10 (183 photos)
MY Y!
FlickrBlog | About Flickr | Community Guidelines | Terms of Use | Privacy Policy | Copyright/IP Policy | Report Abuse
a YAHOO! company

See all public photos tagged with **manchester**

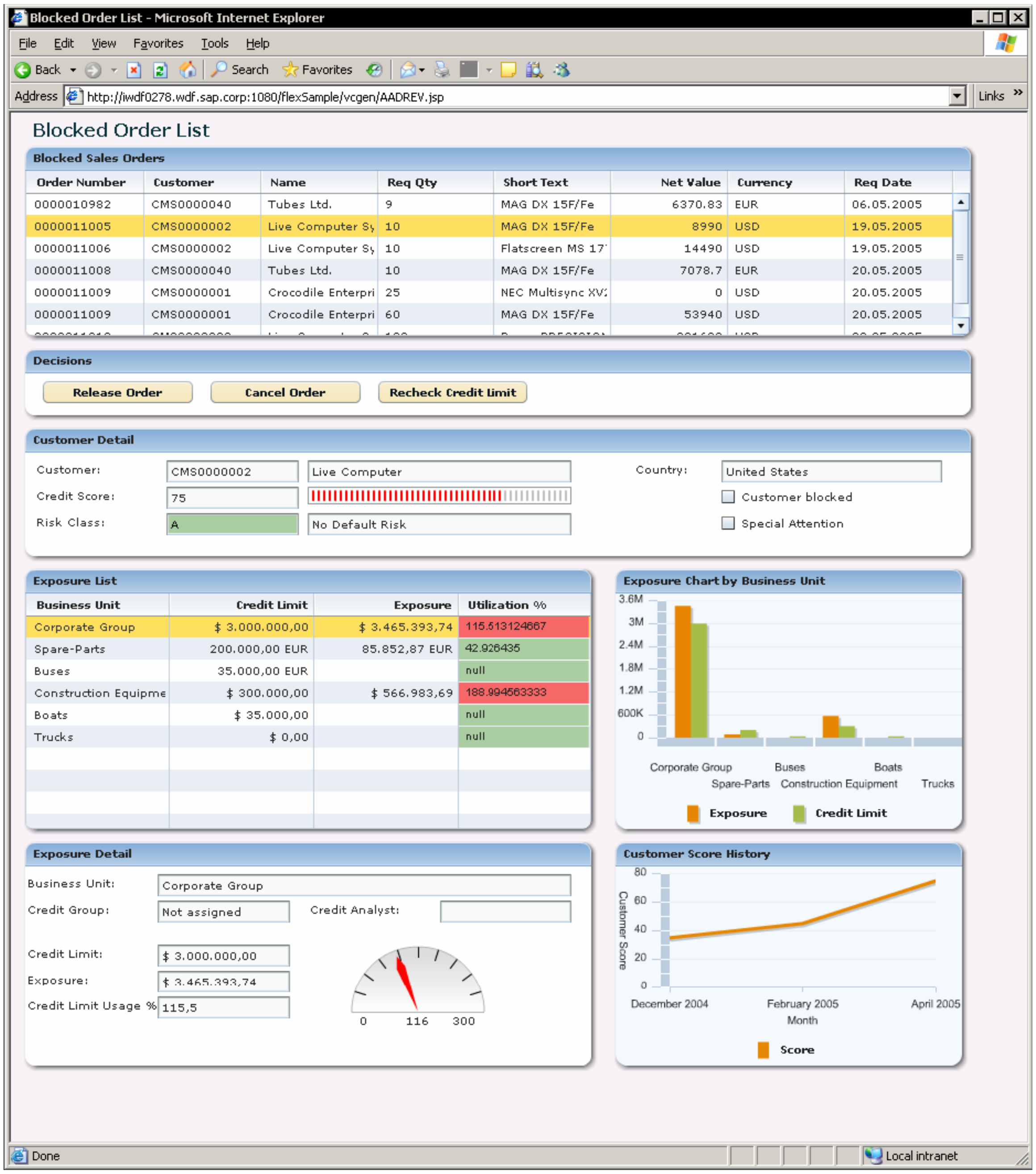
What are tags?
You can give your photos a "tag", which is like a keyword. Tags help you find photos which have something in common. You can assign as many tags as you wish to each photo.

Feeds for Trevira's photos tagged with manchester Available as RSS 2.0 and Atom

Copyright © 2006 Yahoo! Inc. All rights reserved.

Figure 2.6: Flickr [Flic05]

- **SAP Analytics Application "Sales Order Credit Check":** SAP Analytics Application greatly helps to solve business problem in an enterprise. By combining the SAP NetWeaver integration platform and SAP NetWeaver Visual Composer with the Adobe Flex the end-user experience are empowered [CMSW05].


Blocked Order List - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://iwdf0278.wdf.sap.corp:1080/flexSample/vcgen/AADREV.jsp
Links
Done
Local intranet

Blocked Order List

Blocked Sales Orders

Order Number	Customer	Name	Req Qty	Short Text	Net Value	Currency	Req Date
0000010982	CMS0000040	Tubes Ltd.	9	MAG DX 15F/Fe	6370.83	EUR	06.05.2005
0000011005	CMS0000002	Live Computer Sy	10	MAG DX 15F/Fe	8990	USD	19.05.2005
0000011006	CMS0000002	Live Computer Sy	10	Flatscreen MS 17	14490	USD	19.05.2005
0000011008	CMS0000040	Tubes Ltd.	10	MAG DX 15F/Fe	7078.7	EUR	20.05.2005
0000011009	CMS0000001	Crocodile Enterpri	25	NEC Multisync XV	0	USD	20.05.2005
0000011009	CMS0000001	Crocodile Enterpri	60	MAG DX 15F/Fe	53940	USD	20.05.2005

Decisions

Release Order | Cancel Order | Recheck Credit Limit

Customer Detail

Customer:	CMS0000002	Live Computer	Country:	United States
Credit Score:	75			☐ Customer blocked
Risk Class:	A	No Default Risk		☐ Special Attention

Exposure List

Business Unit	Credit Limit	Exposure	Utilization %
Corporate Group	$ 3.000.000,00	$ 3.465.393,74	115.513124667
Spare-Parts	200.000,00 EUR	85.852,87 EUR	42.926435
Buses	35.000,00 EUR		null
Construction Equipme	$ 300.000,00	$ 566.983,69	188.994563333
Boats	$ 35.000,00		null
Trucks	$ 0,00		null

Exposure Chart by Business Unit

Exposure Detail

Business Unit:	Corporate Group		
Credit Group:	Not assigned	Credit Analyst:	
Credit Limit:	$ 3.000.000,00		
Exposure:	$ 3.465.393,74		
Credit Limit Usage %	115,5		

Customer Score History

Figure 2.7: Running the SAP Analytics Application “Sales Order Credit Check” [CMSW05]

Architecture of RIA

The typical architecture for an RIA is shown in Figure 2.8. XML is generally used as the data transfer format and is sometimes also used to describe form layouts. In many instances, the client can stay connected to the data source, so a server can update the client in real time. In this case, the green part "Application Controller/Gateway" should make use rich client technology mentioned later.

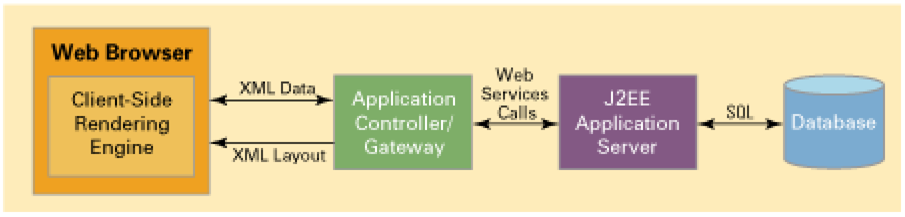


Figure 2.8: Typical RIA Architecture [Rour04]

Requirements for Rich Internet Applications

So to deliver successful Rich Internet Application there are three fundamental elements needed: Rich Client Technology, Server Technology and Development Tools.

- **Rich Client Technology**

1. Rich client technology is the first key element to make it possible to build Rich Internet Applications by providing a runtime environment for deploying rich user interfaces. The client-side applications connect to existing application server backend using asynchronous client/server architecture. This is secure, scalable, and well-suited to the new service-oriented model driven by the adoption of web services. The step of adopting rich client technology will extend capability of browsers and devices for more effective user interfaces.

2. Macromedia RIA Whitepaper [Alla02] announced most crucial principles which Rich client technology should follow:

- *Integrate content, communications, and application interfaces into a common environment.*

  This means, rich clients should provide deep integration for textual content, communications functions and user interface.

- *Provide an efficient, high-performance runtime for executing code, content and communications.*

  This means, rich clients should realize the limitation of current HTML-based web application: slow response time, the lack of client-side data storage; the inability to easily invoke and use remote business logic and improve them.

- *Provide powerful and extensible object models for interactivity*.

  This means, rich clients need to provide a powerful, object-based model for applications and events. This common object model must integrate user interface, communications, and system level services.

- *Enable rapid application development through components and re-use*.

  This means, rich clients should support powerful component-driven development, such as providing extensible components, enabling developers to use third party components and so forth.

- *Enable the use of web and data services provided by application servers*.

  This means, rich clients should provide a model for easily using remote services provided by back-end components, whether hosted in an application server or accessed as XML web services.

- *Embrace connected and disconnected clients.*

  This means, rich clients must enable both types of applications (persistent connections and occasional connections) to be easily built and deployed.

- *Enable easy deployment on multiple platforms and devices.*

  This means, rich clients must support all popular desktop operating systems, as well as the emerging device platforms such as smart phones, PDAs, game consoles, and Internet appliances.

- **Server Technology**

  As the connection between rich clients, application logic and data, The server technology offer a rapid scripting environment, enterprise integration, client connectivity, and support for key standards.

- **Development Tools**

  Having a client and server technology would be much less meaningful without a set of easy and powerful development tools that delivery advanced solutions and allow one to get started quickly. Rich Internet Applications need a range of cooperative development tools due to the client/server architecture.

2.3 RIA Solutions

Among RIA solutions here we focus on such major technologies and their features respectively as:

- Java – Extending rich client
- Ajax Web Framework
- XUL (UI)
- Avalon (now Presentation Framework)
- Adobe Flex 2.0

2.3.1 Java - Extending Rich Client

Java can be used to create almost any rich client imaginable thanks to its solid standards base, homogeneous technology, flexible choice in high-quality tooling. It offers such features as: reliability, availability, scalability security and wide platforms.

Java has also very comprehensive support for building form-based UIs. Besides Java Foundation Classes where the user interface component could be found, there are amounts of standard widget toolkits (SWT) and a raft of other third-party libraries for desktops as well as mobile devices. One can deploy applications by using either Java Plug-In software with a Web browser as an applet or with the newer Java Web Start technology included with the JRE (Java Runtime Environment).

The main disadvantage of using Java to build RIA is its complexity (even simple forms and graphics require dozens of lines of nontrivial code). Its advantages include Java's comprehensive support for web standards and the depth of both the language and its class library. Java RIA offerings typically follow a thin client approach. What involved issues in using Java to build RIAs are the JREs (Java Runtime Environment) supported, the UI libraries employed, and the leverage of Java standards.

2.3.2 Ajax – related Web Frameworks

Definition

Ajax stands for Asynchronous JavaScript CSS DOM and XML and it represents a new web application model. The name was firstly used by Adaptive Path's Jesse James Garrett in his article on AJAX [Garr05]. Particularly, he addressed the usage of "XMLHttpRequest" object for performing asynchronous communications as a key enabler. Since then, the term "AJAX" has spread around the world.

At the first glance at its name, it's not comprised of a single technology, but a combination of cooperative technologies that can be implemented in a more efficient way: XHTML and CSS used for standard-based Presentation; DOM (Document Object Model) used for dynamic display and interaction; XMLHttpRequest used for asynchronous data retrieval [McLe05]; XML and XSLT [Kay01] used for data interchange and manipulation; JavaScript [Croc01] used to bind everything together.

Examples

Gmail [Gmai05], Google Map [Gmap05] and Google Suggest [Gsug05] are in argument as the most popular Ajax based RIA on the Web now. From these applications one can have hints on possibilities realized by using Ajax. It improved the web to be more interactive, more responsive, and smarter than ever while no "click, wait, and refresh" approach anymore.

Advantages

The Ajax model alters classic Web model in two fundamental aspects:

1. **"Partial screen update" replaces the "click, wait, and refresh" user interaction model** [Wei05].

   During user interaction within an AJAX-based application, only user interface elements that contain new information are updated; the rest of the user interface remains displayed without interruption. This "partial screen update" interaction model enables continuous operation context and non-linear workflow possible.

2. **Asynchronous communication replaces "synchronous request/response model"** [Wei05].

   For an AJAX-based application, the request/response can be asynchronous from server interaction. Therefore, the user can continue to use the application while the client program requests information from the server in the background. When new information comes in, only the related user interface will be updated. Figure 2.9 and Figure 2.10 draw a clear distinction between the synchronous model of the classic web application and the asynchronous model of application on top of Ajax.

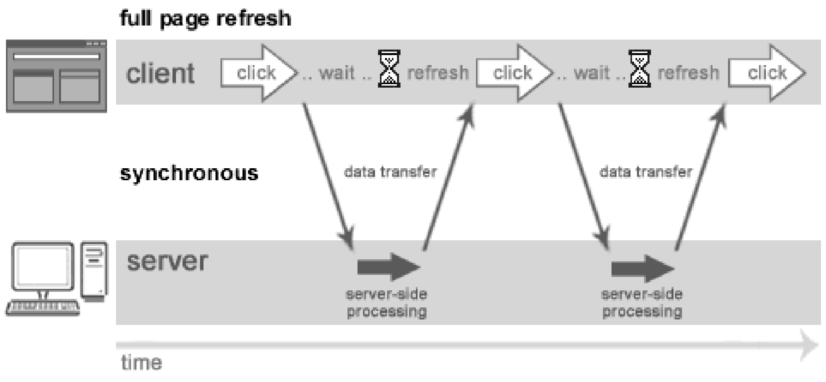


Figure 2.9: Classic Web Application Model: Full page refresh and Synchronous Communication [Wei05]

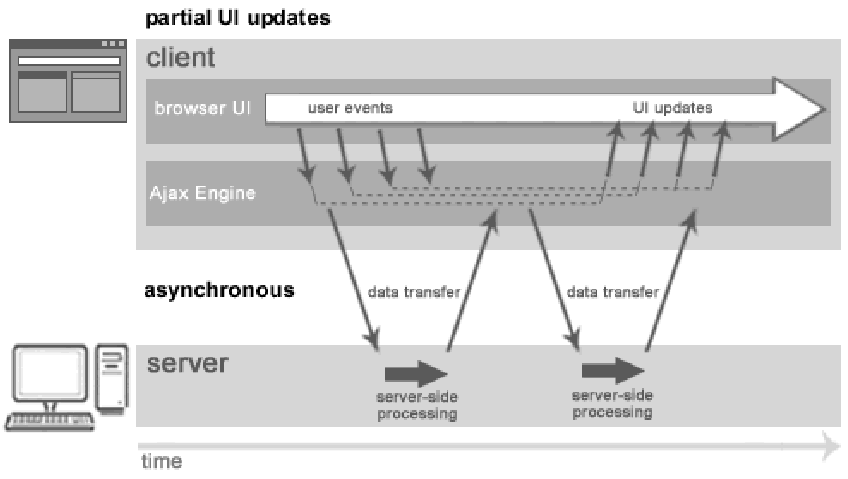


Figure 2.10: Ajax Model: Partial UI Updates and Asynchronous Communications [Wei05]

Architecture

At the view of Software architecture Ajax Framework differs from classic Web application architecture by adding an Ajax engine that can be shown in figure 2.11 below.

While running within a web browser of client-side, this Ajax engine enables the browser to perform a "partial screen update" instead of a "full page refresh" and also communicates with the server-side in the background, decoupling user interaction from server communications.

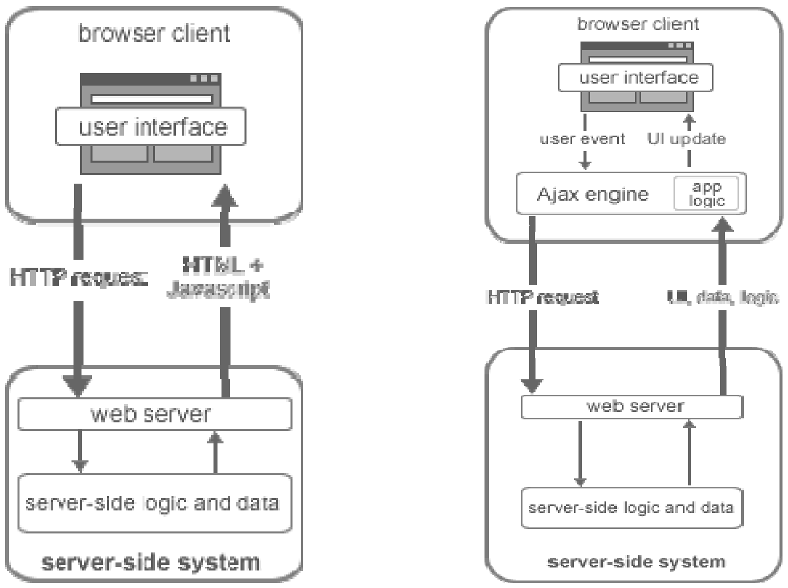


Figure 2.11: Classic Web Application Architecture vs. Ajax Architecture [Wei05]

Limitations

Though Ajax based products appear to be intelligent, it has some unexpected limitations regarding several aspects:

- JavaScript implementations have not been standardized across browsers and operating Systems. As a result issues like browser, operating system, version supporting might be a problem

- JavaScript supports only limited RIA functions. Compared with UI toolkits for desktop applications the rich UI widgets which can be used by Ajax application are relatively poor.

- JavaScript is difficult in maintaining and the performance, security limitations are also challenging in case building large scale Internet applications.

2.3.3 XML UI

XML UI defines an innovative technique meaning User Interface described by XML. Some of following Frameworks like Adobe Flex, Avalon, are generally based on this idea. This approach is superior to API-based UI toolkits such as Java Swing because it clearly separates the user interface into four parts: content, which is the structure and description of UI elements; appearance, which defines the look and feel of an application; behavior and localization information for internationalization.

Resource

There are a lot of Open Source XML UI Toolkits in Java [XmlU05]. These frameworks can save developer much of the code typically needed to build an application. The results are savings in development time and maintenance costs and greater stability.

XUL Definition

XUL stands for XML UI Language and was pioneered by Mozilla. It is used to create forms applications running in the Mozilla browser as well as other rendering engines. As with Java, there is a fairly large user community with plenty of open source tools such as the Theodore Thinlet Editor, a Java application that allows one graphically to lay out a UI and generate the corresponding XUL [Rour05].

Advantage and disadvantage

XUL's advantages include: it has open access to a number of Web standards because it integrates with the Gecko engine; XUL is very expressive and compact language compared with other XML UI description languages. As a disadvantage it lacks support by a major commercial entity.

2.3.4 Windows Presentation Foundation (formerly code name "Avalon")

Introduction of Windows Vista

Longhorn (now Windows Vista) is the code name for the next major Windows desktop operating system release to follow Windows XP of Microsoft. The scheduled release date for Longhorn has been pushed back several times; the operating system is currently scheduled for release sometime in the latter part of 2006.

Overview of WinFX

From a programmer view of Longhorn, its new programming model names WinFX and it consists of three components as figure 2.12 shows: Avalon, Indigo, WinFS which represent the presentation layer, the networking Web service layer and the storage layer respectively.

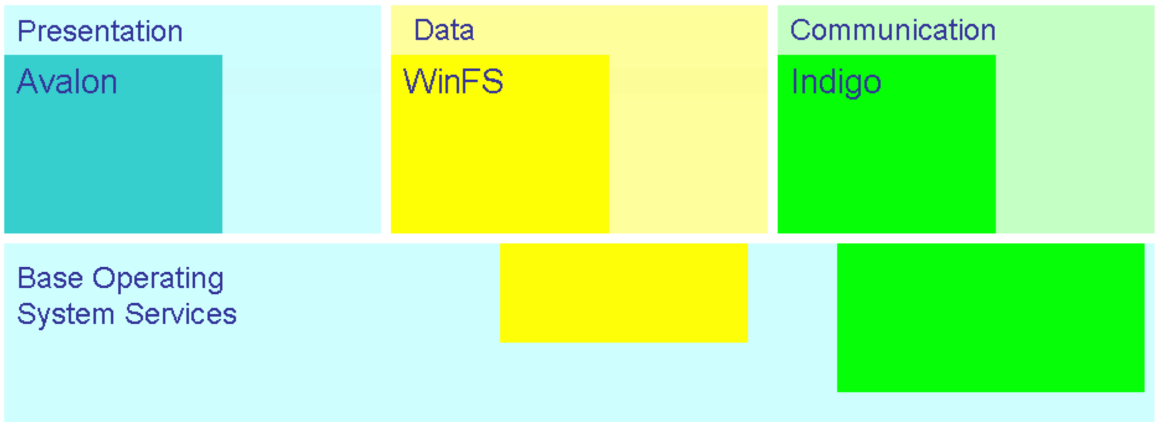


Figure 2.12: WinFX [Long06]

Overview of Avalon

As the core portion of presentation layer in Longhorn, Avalon (now Windows Presentation Foundation) integrates Document, UI elements, Media to provide the foundation for building Windows applications, Web applications, graphical applications, and applications running on various devices. The architecture of Avalon is demonstrated in figure 2.13.

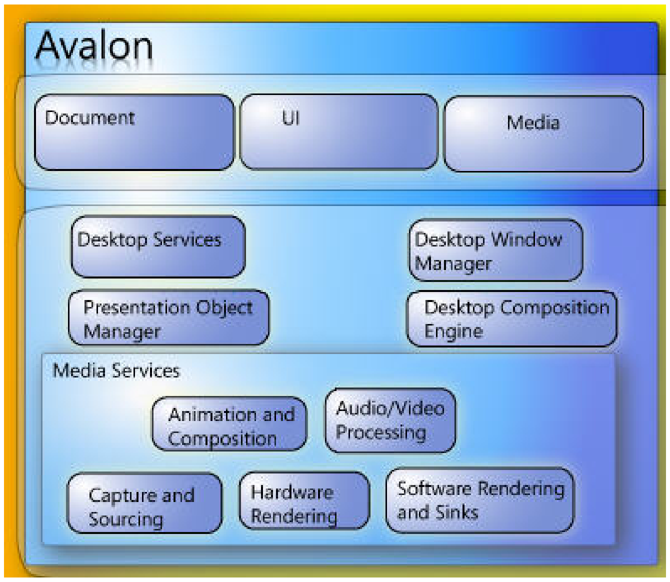


Figure 2.13: Avalon Architecture [Long06]

Avalon (media) integrates both 2-D and 3-D graphic, seamless management of audio files, features to play video, animated controls and other visual elements. Avalon (Document, UI elements) supports fixed flow and adaptive layouts, pagination as well as printing. Because Avalon supports 3-D rendering and the rendering is based on vector graphics, the hardware requirement is high, especially an appropriate graphics card or updating previous graphic card is commonly needed.

XAML

- XAML is an XML-based markup language used in Avalon to create User Interface. It based on the approach of XML UI mentioned in section 2.3.3. XAML corresponds to the underlying .NET object model.

- Simplest example: Hello World

```
<DockPanel xmlns="http://schemas.microsoft.com/2003/xaml">
   <Text>Hello World</Text>
</DockPanel>
```

  The code above shows that XAML is just XML with a defined schema. The tags correspond to objects and attributes correspond to properties.

2.3.5 *Adobe Flex 2.0 (formerly Macromedia Flex)*

Background of Flex

As mentioned above, the phrase “Rich Internet Application” was first addressed by Macromedia in early 2001 after seeing that developers and designers were building rich Internet application in Macromedia Flash. Flash is arguably (depending on the Flash Player version) the most widely deployed front-end technology on the Web, claiming up to 98 percent penetration across all desktops.

The first complete family of product and technologies dedicated to work together to deliver Rich Internet Applications is Macromedia MX product, which was announced at August 2003.

In March 2004 Macromedia released their presentation server and application framework Flex 1.0, which enables enterprise development teams to put more effective interfaces on serious business applications. Flex represents new application architecture. This new approach blends the flexibility of services-oriented data access with the superior reach and effectiveness of a cross-platform rich client [Flex04].

The announcement of Flex 2.0 is on 2005, almost one year after the release of version Flex 1.5. The facts reflex that Macromedia is continuously engaging in developing comprehensive RIA solutions.

Overview of Flex 2.0

Flex 2.0 is actually a new lineup of developer tools, libraries, and runtime services that will enable developers everywhere to build and deploy Rich Internet Applications that take advantage of the Flash Player runtime. It also uses an XML-based language called MXML that provides a declarative way to manage the visual elements of an application. The framework of Flex 2.0 offers an extensible and customizable class library of pre-built components, effects, behaviors, and layout managers. Flex compiler is now a part of IDE - Flex Builder 2.0 (Eclipse based tools) and executes Flex application locally. Therefore the Flex applications no longer need compiler in the server side [Flex05].

Foundation of Flex 2.0

The product line of Flex 2.0 is based on Flash Player 8.0 (scheduled for release in spring 2006, currently in public beta) and ActionScript 3.0.

- Flash Player 8.5

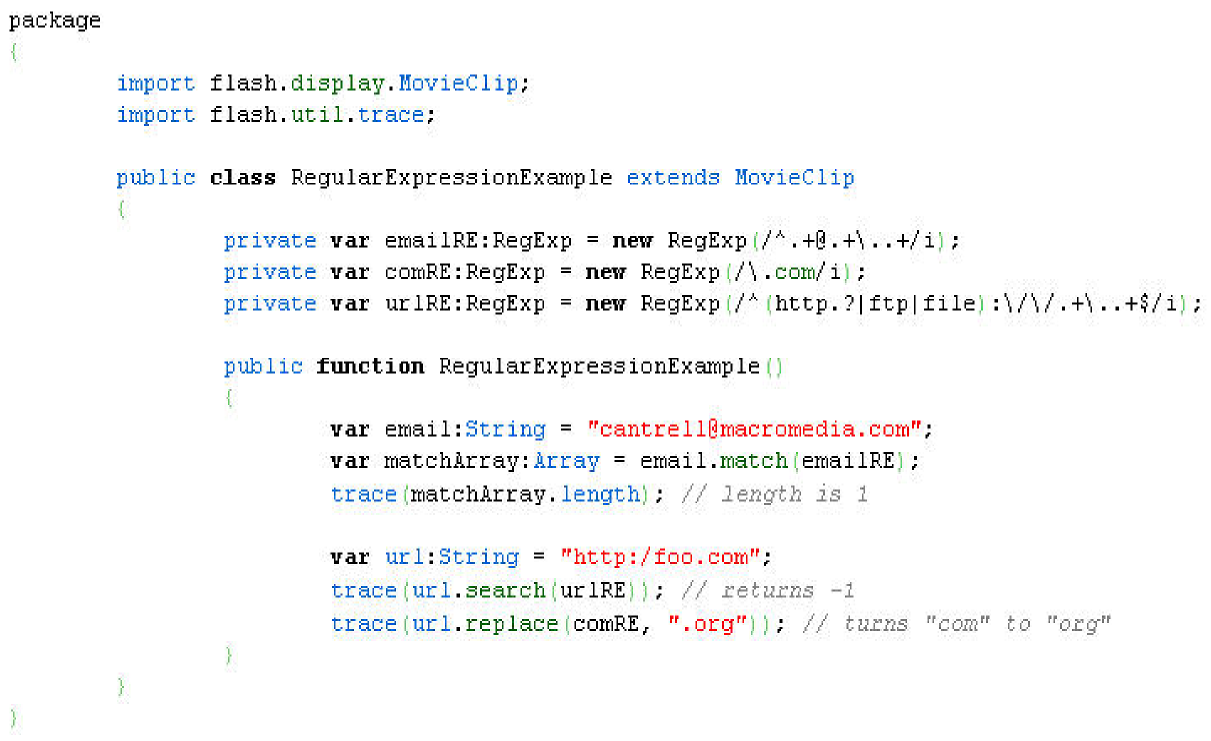
  Flash Player is the backbone of Flash Platform. Its primary function is to be as a client for playing animation and support for an embedded scripting language

ActionScript (AS), which is based on ECMAScript (the same standard that drives the development of JavaScript). Adopting the advanced of Flash Player 8.0, version 8.5 provides great performance not only in the rendering engine, video supported, and enhancing API, but also improves script execution in the virtual machine. The new virtual machine is known as AVM2 and promised to be significantly faster, support full runtime error reporting and industry-standard debugging. For backward-compatibility reasons AVM1 which executes Action Script 1.0 and 2.0 codes is included [Wadh05].

- Action Script 3.0

The first version of Action Script is a primitive script language which didn't even support variables. Over the years AS has evolved with the update of Flash Player to an object-oriented programming language.

Action Script 3.0 based on the next generation of ECMAScript standard and contains a host of new features. The support of Regular expression enables better operations on text: parsing, validating, processing. ECMAScript for XML (E4X) transforms XML into a native data type, thus manipulating XML becomes more natural. The standardized DOM event model gets more unified. Stronger compile-time type checking is supported, and so forth. The code below shows how to implement a validating class using RegExp.



```
package
{
        import flash.display.MovieClip;
        import flash.util.trace;

        public class RegularExpressionExample extends MovieClip
        {
                private var emailRE:RegExp = new RegExp(/^.+@.+\..+/i);
                private var comRE:RegExp = new RegExp(/\.com/i);
                private var urlRE:RegExp = new RegExp(/^(http.?|ftp|file):\/\/.+\..+$/i);

                public function RegularExpressionExample()
                {
                        var email:String = "cantrell@macromedia.com";
                        var matchArray:Array = email.match(emailRE);
                        trace(matchArray.length); // length is 1

                        var url:String = "http:/foo.com";
                        trace(url.search(urlRE)); // returns -1
                        trace(url.replace(comRE, ".org")); // turns "com" to "org"
                }
        }
}
```

Figure 2.14: Regular Expression Example [Regu06]

Involved Technologies in Flex 2.0

Based on Flash Player 8.5 and Action Script 3.0 other technologies of Flex 2.0 Product line need to be talked about are Flex Framework 2.0, Flex Builder 2.0, Flex Enterprise Services 2.0, Flex Compiler and Flex Charting Components 2.0.

- Flex Framework 2.0

  Flex Framework 2.0 is the core of Flex 2.0. It can be used to build and style Flex applications without a server or any particular IDE. The major pieces of it are programming languages (MXML and Action Script 3.0), core application services and class library, components, compiler and tools.

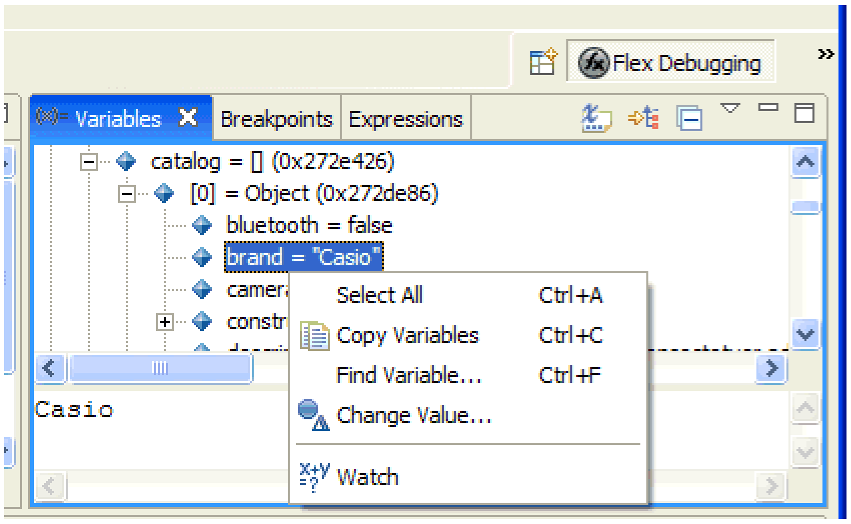
  Framework 2.0 adds a rich set of class library including extensible UI components, a flexible model for controlling layout and user interaction, and a robust infrastructure for data binding.

  More specifically, the component library Flex 2.0 provides more rich components (visual components and service components) than before. Support of Rich Text Editor is an example. One doesn't have to make it additionally using Flash authorities tools and import the executed SWF into Flex like in 1.5 anymore. Developers can directly use these components, subclass them to modify their behaviors or use component API to create new components. Using MXML to layout and control an application's visual look is a flexible approach. The robust data binding infrastructure enables automatically updates the user interface when data is returned, allowing applications to remain responsive even when waiting for results from the server and never lost the context ("Partial screen update").

- Flex Builder 2.0

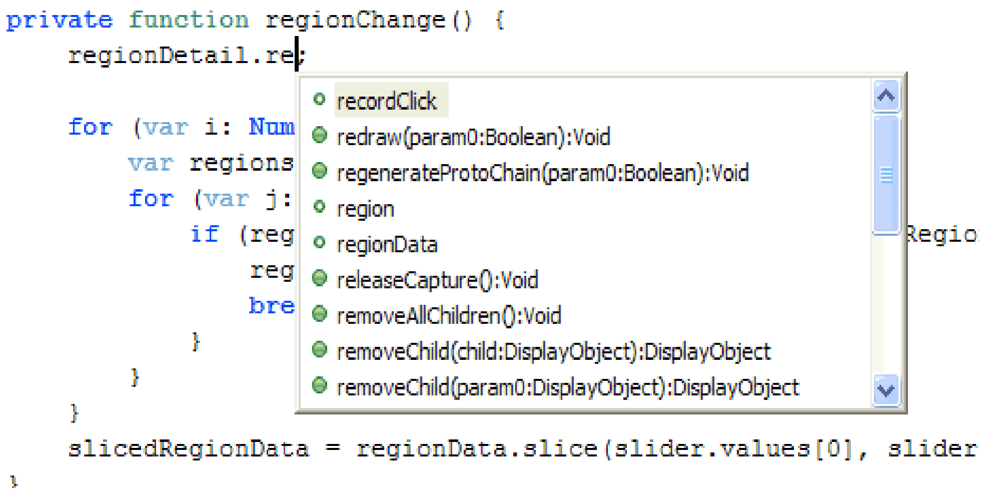
  Based on the Dreamweaver MX 2004 the previous authoring tools for Flex 1.5 is Flex Builder 1.5. It can provide things like design and code view, syntax highlighting, simple code hinting, and application preview.

  Flex Builder 2.0 has almost nothing in common with Flex Builder 1.5 because it is build on top of Eclipse. Currently it can be available as a standalone application or as an Eclipse plug-in that developers can use with theirs existing Eclipse installation. Features like integrated compiler, code hinting, debugging, and design view, source control system integration all make development much simpler. Figure 2.15, figure 2.16 show how these features look like in Flex Builder 2.0.



```
catalog = [] (0x272e426)
  [0] = Object (0x272de86)
    bluetooth = false
    brand = "Casio"
    camer
    constr
Casio
```

Figure 2.15: Flex Debugger [FBui06]



```
private function regionChange() {
    regionDetail.re
    
    for (var i: Num
        var regions
        for (var j:
            if (reg                                                      Regio
                reg
                bre
            }
        }
    }
    slicedRegionData = regionData.slice(slider.values[0], slider
}
```

```
recordClick
redraw(param0:Boolean):Void
regenerateProtoChain(param0:Boolean):Void
region
regionData
releaseCapture():Void
removeAllChildren():Void
removeChild(child:DisplayObject):DisplayObject
removeChild(param0:DisplayObject):DisplayObject
```

Figure 2.16: Code Navigation [FBui06]

- Flex Enterprise Services 2.0

  Flex Enterprise Service 2.0 includes three components: Message Service, RPC Services and Data Services which are shown in figure 2.17.

  Flex Message Service is the foundation of the Flex Enterprise Services and provides publish/subscribe messaging infrastructure. It is made of two key components: a message service running in the application server, and a client-side

API. The Flex Data Services automatically synchronizes data manipulated locally with data on the server.

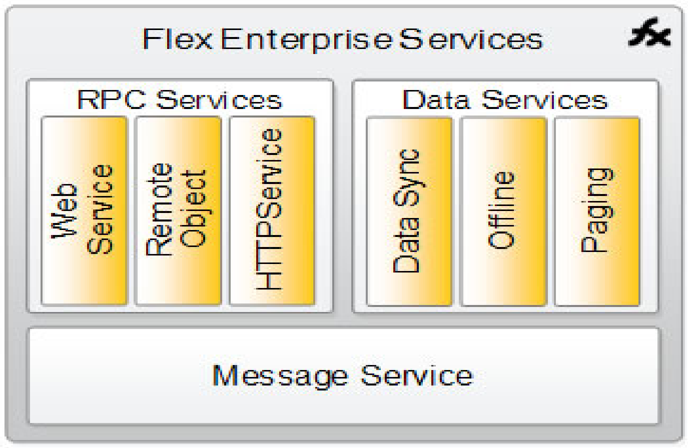


Figure 2.17: Flex Enterprise Services Architecture [Fent06]

The Flex Enterprise Services enable developers to build “Data-Rich” Internet Applications that are not only rich in terms of the user interface, but also in terms of how the data flows between tiers [Coen06].

- Flex Compiler

  Although Flex Builder 2.0 and Flex Enterprise Services 2.0 both have the Flex compiler built in, it can also be used outside of either product. From the command line the compiler can also be used for compiling MXML or ActionScript applications.

- Flex Charting Components 2

  The Flex Charting Components 2 is an extension to the Flex Framework and provides very slick data visualization capabilities. Developers can use them with Flex Builder 2 and Flex Enterprise Services.

Flex Architecture

Architecture is an important issue before building any application. Rich Internet Applications have the same requirements as any other applications: stability, robustness, reusability and extensibility. The proven software engineering methodologies and practices is well-suit for Flex based application too.

MVC architecture

MVC architecture is the well common used design patterns. It promotes the overall maintainability of an application and the reusability of classes. Using MVC architecture the entire application will be partitioned into three categories of classes:

- Model classes: classes that encapsulate data and behaviors related to the data;
- View classes: classes that are responsible for the user interface;
- Controller classes: classes, those are responsible for the plumbing.

Loose Coupling of Application Components

Loose coupling of components is another best practice in object-oriented development. Loose coupling is a programming technique that consists in avoiding interdependencies between classes, so that it can increase the reusability of components throughout an application and across applications.

In Flex based applications it' better to use event notifications between components and avoid direct reference to other components.

Besides these there are many other design patterns and best practice can be applied to Flex Applications. But it doesn't mean the more design patterns used in an application, the more better the application works. Capturing the requirements of an application and then applying appropriate design patterns are essential.

Cairngorm Framework

Cairngorm is an open-source architectural framework which can be used as a skeleton for development of RIAs. It was released by a software consultancy: iteration::two [Iter05]. Cairngorm Framework borrows a small number of relevant design patterns from the core J2EE Patterns advocated by Sun Microsystems and reassembles them in order to provide microarchitecture for the declarative programming model of Flex. The fundamental patterns introduced in the Cairngorm architecture are the Value Object pattern and the Model Locator pattern. Three key areas addressed by Cairngorm are [Webs06]:

- Handling user gestures on the client
- Encapsulating business logic and server interactions
- Managing state on the client and representing this state to the user interface

2.4 Comparison

The comprehensive introductions in section 2.3 coved the current popular RIA solutions: Java, XML UI, Ajax-related web framework, Avalon, and Adobe Flex framework.

Basically, these approaches all fulfill RIA technologies requirements (see section 2.2) and have successful examples in different fields. Substantially, they vary in their technical foundation and their suitability for specific problems. Therefore, an objective evaluation will be helpful in choosing an appropriate approach to build web application. Table 1 uses several essential factors from the aspects of business and technology like cost, functionalities provided, and richness of UI etc. to differentiate these technologies.

	Java	**XUL**	**Ajax**	**Flex 2**	**Avalon**
Cost	Open source	Open source	Open source	Commercial product, Flex Builder 2 will be sold for less than $1000 per developer.	Commercial product, cost money
Mature functionality	Reliability, availability, scalability and security	very mature	Performance and functionality are limited due to JavaScript	Mature application framework both in user experience and code maintainability	Comprehensive functionalities
UI richness	high-level libraries (Swing, JFace)	a powerful set of user interface widgets	Available rich visual components are poor and only supports parts of all RIA functions	as rich as traditional desktop applications	abandon visual components provided
Media supported	Java Media Framework API.	Limited supported	Supported only through external plug-ins (like Media Player).	Dynamically load audio, embedded flash audio	Powerful media services embedded

Browser Integration	as applet in a browser	mainly running in Mozzila	JavaScript natively supported by modern browsers.	Flash Player plug-in required.	No plug-in needed
Browser Compatibility	Cross platforms	Cross platforms	Major compatibility differences between browser versions.	Cross platforms	Cross platforms
Programming Model	Java	XBL	JavaScript, XML	MXML and Action Script 3.0	XAML
Server integration	Many solutions available.	Limited. With PHP server	Limited. Dynamically with server using XMLHttpRequest Object.	Enterprise Services 2 provides powerful mechanism	Many solutions available.
XML supported	full support	full support	full support	full support	full support
Development tools	Eclipse	plenty of open source tools such as the Theodore Thinlet Editor	No ideal development tools available currently	Flex Builder 2.0 can be available as a Eclipse plug-in.	Visual Studio .NET IDE

Table 1: Comparison of RIA Technologies

From the final comparison of their features Flex 1.5 (Flex 2.0 was not released at that time) was chosen here to perform implementation because it fulfils all requirements of the *Social Organizer* and thus provides user real breakthrough visual effects and comprehensive rich functionalities.

3 Social Organizer UI Analysis and Design

*This chapter explains the concept of **User-Centered Social Software** and describes the developing process of the prototype system **Social Organizer** in detail: the architecture, the information model and the implementation of the UI. After reading this chapter it is clear that how Flex Framework discussed in the previous chapter benefits the application Social Organizer in many ways.*

3.1 Background of User-Centered Social Software

On the Internet, several types of multi-user platforms exist that can be accessed by users role-based and device-independently. Besides centralized software systems like enterprise information portals, in which all information is managed at a single location, individual users increasingly become members of several community platforms like social software (section 3.1.2) that focus on information search, publication and sharing support for their members. Thus, personal information is distributed across multiple platforms in addition to personal devices and applications [LMW05]. The idea behind **User-Centered Social Software** is therefore to provide a unified view on all available information and services in the social context of the user. Services of social software are integrated for supporting use cases like searching, sharing and organizing information across multiple platforms [LMW05]. The prototype application of such user-centered social software is referred to as the *Social Organizer* in this report.

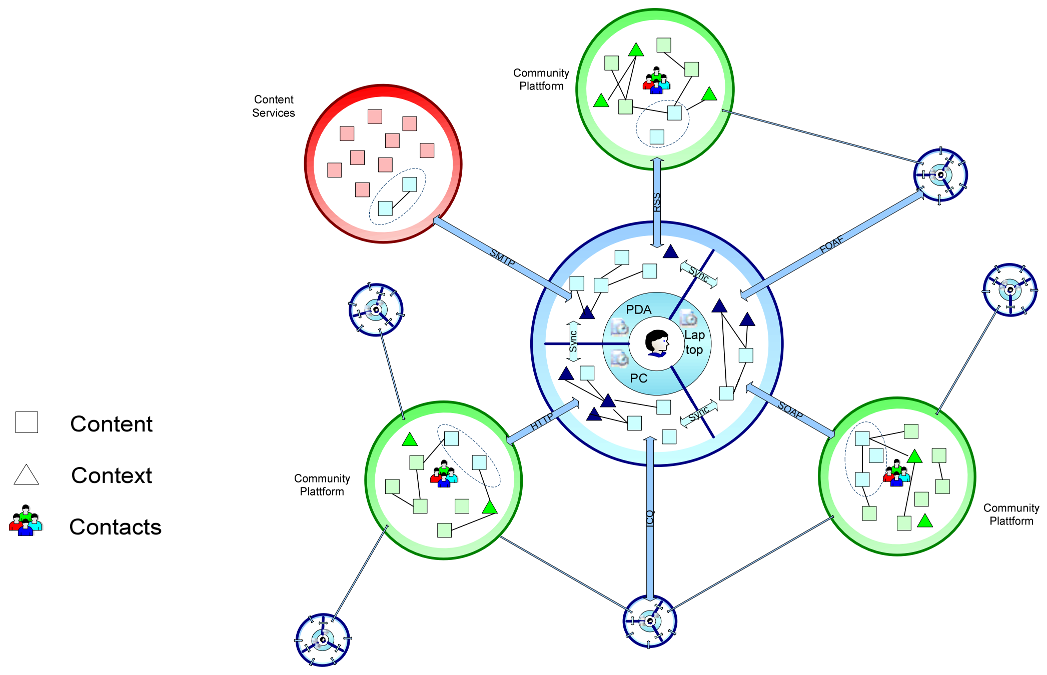


Figure 3.1: Scenario for User-Centered Social Software [Lehe05]

Figure 3.1 illustrates this idea more clearly. People store and manage personal information in various devices (PDA, Laptop, and PC etc.). They are also members of different community platforms and share some contents there. Different contents are linked together through the same social context. Contacts are built up because context groups people in a community. All of the information, which is located inside or outside, needs to be well organized and synchronized with each other. Therefore, as an extension of personal information management (PIM) and client software on the personal device the *Social Organizer* enables user to organize information in a social way that is in the social context of the user.

3.1.1 Social Software

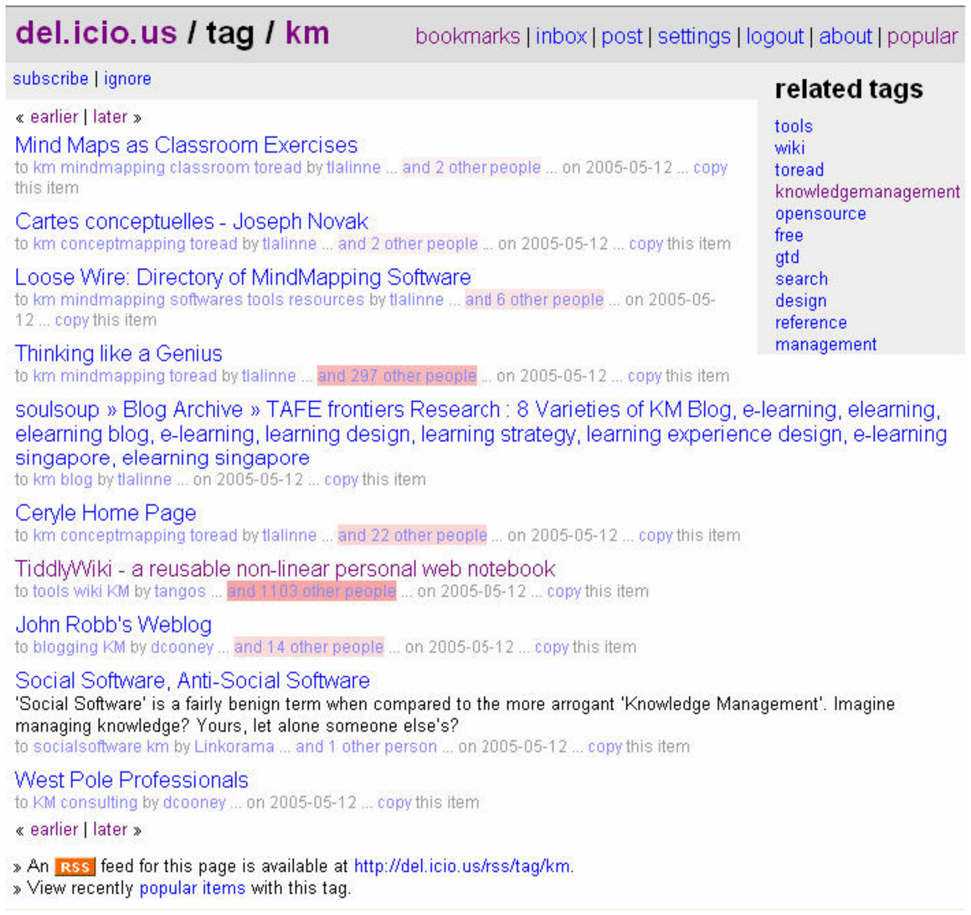
Social Software, which enables individuals and groups to interact, collaborate and form online communities through computer-mediated communications, has become relatively popular during the last years. Weblogs, Social Networking Services and Object-Centered Social Software are the newly emerging classes of this kind of services.

Weblogs make publishing of personal knowledge and information easy.

Social networking services allow people to build up and manage contact networks on a community platform. Well-known communities are LinkedIn [Link06] in the USA and OpenBC [Open06] in Europe. New contact or relationship of members from different communities can be set up by means of viewing the path of indirect contacts.

Object-Centered Social Software provides users the opportunity to perform series of activities like to store, organize, and share their individual collections of objects. Objects can be different types of contents: documents, bookmarks, references, multimedia files etc. Users can freely choose keywords (tags) to classify their collections into different categories. The keywords chosen by a member represent shared metadata and context information about this user in the community [Lehe05]. Well-established examples are Del.icio.us [Deli05] for bookmark sharing and Flickr [Flic05] for photo sharing. Unlike social networking services, this kind of community platforms try to group people around shared objects, resulting in different kinds of social groups in a community.

Figure 3.2 shows where all bookmarks that are tagged with “km” in del.icio.us. The related tags here “tools”, “wiki”, “toread”, “knowledge management” etc. describe different meanings as “km” by individuals and have more common meaning than “km” in public. It also shows how many other users are interested in the same bookmark and their feedbacks.



del.icio.us / tag / km   bookmarks | inbox | post | settings | logout | about | popular

subscribe | ignore

« earlier | later »

Mind Maps as Classroom Exercises
to km mindmapping classroom toread by tlalinne ... and 2 other people ... on 2005-05-12 ... copy this item

Cartes conceptuelles - Joseph Novak
to km conceptmapping toread by tlalinne ... and 2 other people ... on 2005-05-12 ... copy this item

Loose Wire: Directory of MindMapping Software
to km mindmapping softwares tools resources by tlalinne ... and 6 other people ... on 2005-05-12 ... copy this item

Thinking like a Genius
to km mindmapping toread by tlalinne ... and 297 other people ... on 2005-05-12 ... copy this item

soulsoup » Blog Archive » TAFE frontiers Research : 8 Varieties of KM Blog, e-learning, elearning, elearning blog, e-learning, learning design, learning strategy, learning experience design, e-learning singapore, elearning singapore
to km blog by tlalinne ... on 2005-05-12 ... copy this item

Ceryle Home Page
to km conceptmapping toread by tlalinne ... and 22 other people ... on 2005-05-12 ... copy this item

TiddlyWiki - a reusable non-linear personal web notebook
to tools wiki KM by tangos ... and 1103 other people ... on 2005-05-12 ... copy this item

John Robb's Weblog
to blogging KM by dcooney ... and 14 other people ... on 2005-05-12 ... copy this item

Social Software, Anti-Social Software
'Social Software' is a fairly benign term when compared to the more arrogant 'Knowledge Management'. Imagine managing knowledge? Yours, let alone someone else's?
to socialsoftware km by Linkorama ... and 1 other person ... on 2005-05-12 ... copy this item

West Pole Professionals
to KM consulting by dcooney ... on 2005-05-12 ... copy this item

« earlier | later »

» An RSS feed for this page is available at http://del.icio.us/rss/tag/km.
» View recently popular items with this tag.

related tags
tools
wiki
toread
knowledgemanagement
opensource
free
gtd
search
design
reference
management

Figure 3.2: Del.icio.us [Deli05]

The benefits of this collaborative tagging service are: Individuals can access their own bookmarks, classify contents in social contexts, learn about new content from others' posts feedback, and find people through shared contents. In the network new vocabularies can be established.

3.1.2 Use Cases of UCSS

Figure 3.3 shows a view of User-Centered Social Software as an extension of Personal Information Management. In this section the information workflow with its specific meanings will be explained, which are the fundamental concepts for the implementation of the *Social Organizer* user interface.

Content here contains different types of objects such as text files, RSS feeds, contacts etc. or even notes, annotations that can be used as private or public (to publish on certain social software systems in the future). The context links together content and contacts and defines the way how to share these objects.

According to the technical report on User-Centered Social Software-Beyond Closed Community Platforms in 2005 [LMW05] *Social Organizer* is proposed to support bidirectional information processes. The information flow in figure 3.3 depicts these processes:

The use cases of the *Collect* process are the following: finding a specific content object by its URI, searching content locally or externally using keywords, browsing content, subscribing in order to automatically collect contents, getting feedback of content published on communities, or synchronizing equivalent versions of content automatically.

The *Organize* process means, that the user can classify a content by giving it specific context definition, link a content semantically to another content, store a content persistently on his personal devices, annotate a content, rate the content from his personal view, or create and delete content etc.

The *Share* process describes how a user publishes contents and shares them with contacts in a controlled way.

The *Learn* process will be accomplished in the way the user receives feedback from others and also gives feedback to his contacts.

In section 3.3 some of above mentioned processes are shown in detail.

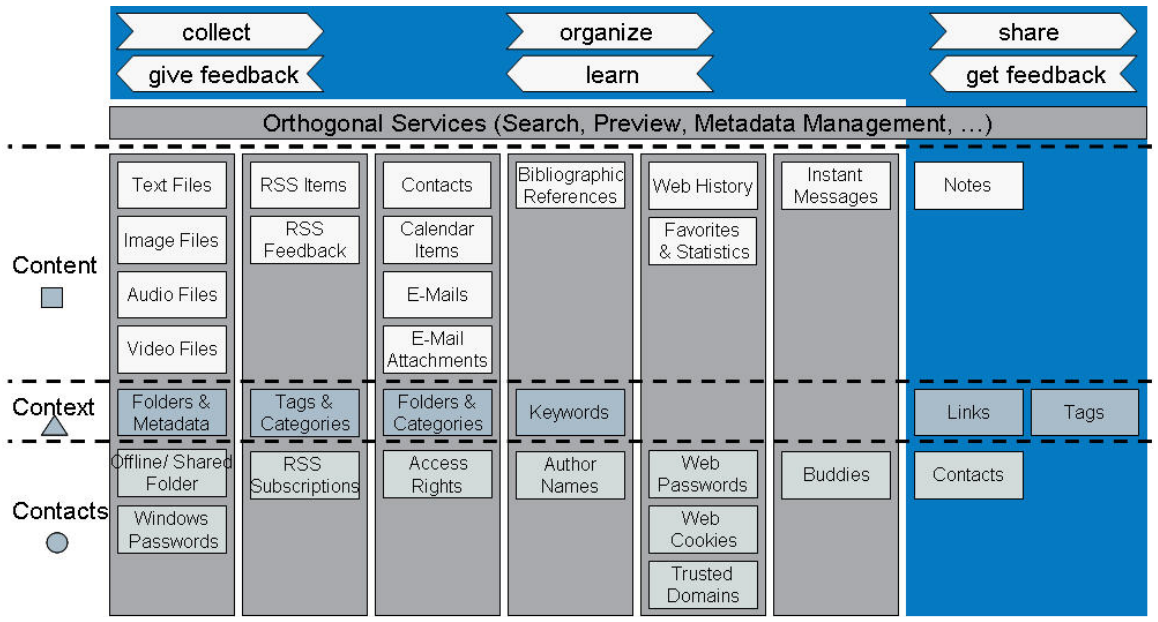


Figure 3.3: User-Centered Social Software as an Extension of Personal Information Management [Lehe05]

3.2 Design Considerations for the Social Organizer

3.2.1 Architecture Overview

Figure 3.4 shows the distributed content syndication architecture of user-centered social software – *Social Organizer*. In this SEP period, the UI of the *Social Organizer* was implemented with Flex 1.5.

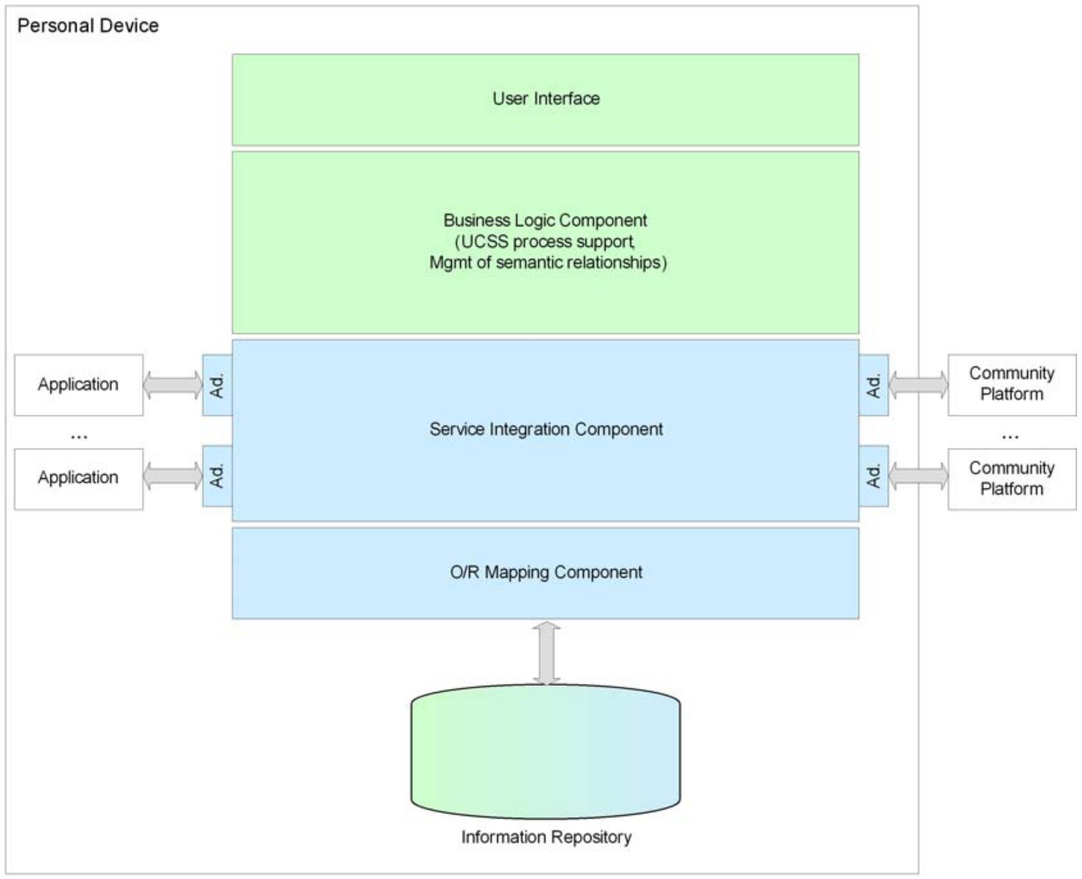


Figure 3.4: Architecture Overview of the *Social Organizer* [Lehe06]

The user interface is designed to allow individuals to manage all their information by personal devices and on social communities simultaneously. In this work most of the User Interface, some of Business Logic Component are involved and implemented as well. As Information Repository the local shared object (see Section 3.3.1) mechanism was used. The information stored in the repository can only be accessed by the local user since it resides on the personal device.

3.2.2 *Information Model*

The information model of the *Social Organizer* is demonstrated in Figure 3.5. According to the requirements of the user interface, the relevant classes have been implemented in the `model` package. These classes are `Annotation`, `CommunityPlatform`, `Contact`, `ContentProvider`, `ContentRef`, `ContentType`, `TaskContext`, `Portfolio`, and `Tag` (with blue border colour). Their association relationships have following meanings:

A `CommunityProvider` can manage a set of `ContentRef` objects each of which has a `ContentType` that can be associated with multiple `ContentRef` instances. The `CommunityPlatform` extends the abstract class `CommunityProvider` and can be associated to several `TaskContext` instances; A `TaskContext` can contain multiple `Portfolio` instances which contain `ContentRef` objects; multiple instances of `Annotation` can be associated with a `ContentRef` object; Concerning context information, multiple `ContentRef` objects can be classified by one `Tag` whereas one `ContentRef` can be given multiple `Tags`;

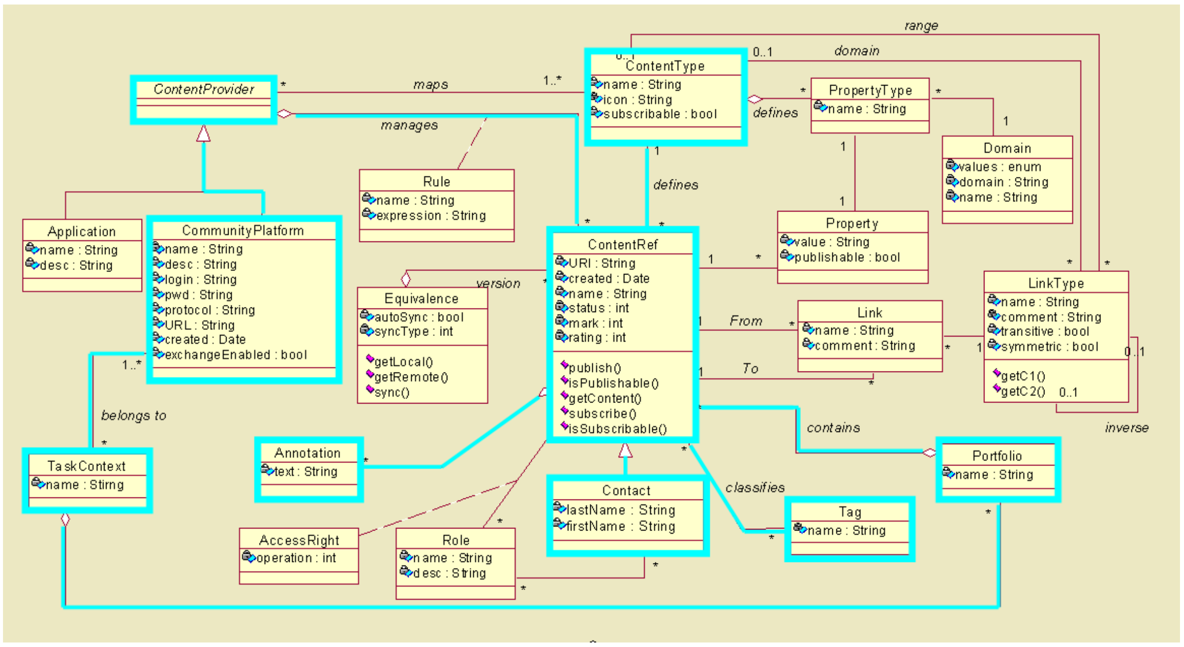
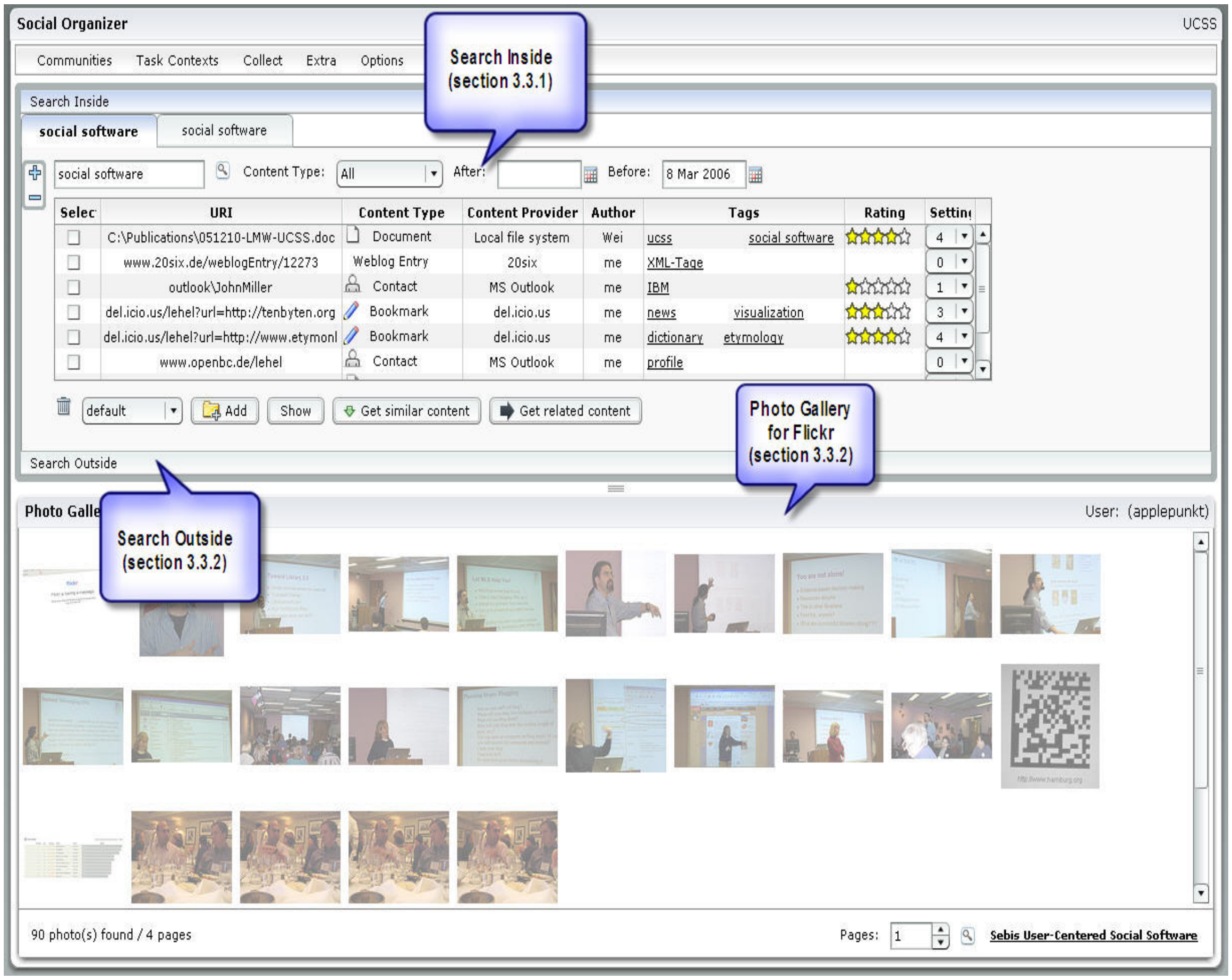


Figure 3.5: Information Model of the Social Organizer [Lehe06]

The functions which perform data manupulation, data management like to read and process xml data, to mapping them into relative model classes, or to retrieve data collection based on various criteria are wrapped explicitly in classes `Annotations.as`, `CommunityPlatforms.as`, `Contacts.as`, `ContentProviders.as`, `ContentRefs.as`, `ContentTypes.as`, `Contexts.as`, `Domains.as`, `Links.as`, `LinkTypes.as`, `Portfolios.as`, `Propertys.as`, `PropertyTypes.as`, `Tags.as`. These AS classes are included in the `dataWrapper` package. Instances of these classes can be used directly to execute data logic. The default data, which the application uses, should come from server side in the future. But in the meantime of UI implementation a set of xml files are proposed to model this process. They are all available in the `serverData` package.

3.3 Implementation of the Social Organizer User Interface



Selec	URI	Content Type	Content Provider	Author	Tags		Rating	Settin
☐	C:\Publications\051210-LMW-UCSS.doc	Document	Local file system	Wei	ucss	social software	★★★★☆	4
☐	www.20six.de/weblogEntry/12273	Weblog Entry	20six	me	XML-Tage			0
☐	outlook\JohnMiller	Contact	MS Outlook	me	IBM		★☆☆☆☆	1
☐	del.icio.us/lehel?url=http://tenbyten.org	Bookmark	del.icio.us	me	news	visualization	★★★☆☆	3
☐	del.icio.us/lehel?url=http://www.etymonl	Bookmark	del.icio.us	me	dictionary	etymology	★★★★☆	4
☐	www.openbc.de/lehel	Contact	MS Outlook	me	profile			0

Figure 3.6: User Interface of the Social Organizer

The user interface of the *Social Organizer* is shown in Figure 3.6. The interface is contained within a Panel layout, adding a MenuBar and a vertical two-panel layout to show the summary of search transactions, and as an example for the extension of search outside a panel containing images searched at Flickr [Flic05].

Implementation of the main panel

As mentioned in the previous chapters, flex adopts this idea, which uses xml to descript and create user interfaces, but with its own xml language (mxml) as described in Chapter 2.

All files that concern visual interface are in the `ucss/view` package except `index.mxml` which assembles all classes together.

The code below is a part of `index.mxml` and used here as an example to give an overview of visual composition.

```
<?xml version="1.0" encoding="utf-8"?>

<mx:Application xmlns:mx="http://www.macromedia.com/2003/mxml"
xmlns:s="view.searchPanel.*"
xmlns:com="flickrAPI.com.macromedia.flickr.*"
xmlns:fui="flickrAPI.ui.*" >

<mx:Script source="index_script.as"></mx:Script>

<mx:Panel id="wholePanel" title="Social Organizer" status="UCSS">

        <mx:Model id="menulist" source="config/menudata.xml"/>
        <mx:MenuBar id="menuB" change="changeItem(event)"
        dataProvider="{menulist.menuitem}" initialize="menuItemInit()"/>

        <mx:VDividedBox>
                <mx:HBox id="searchComp">
                        <mx:Accordion id="search">
                                <mx:TabNavigator label="Search Inside" id="tnIn">
                                        <s:searchIn id="searchPanelIn"/>
                                </mx:TabNavigator>
                                <mx:TabNavigator label="Search Outside" id="tnOut">
                                        <s:searchOut id="searchPanel"/>
                                </mx:TabNavigator>
                        </mx:Accordion>
                </mx:HBox>
                <mx:Panel id="galleryPanel" title="Photo Gallery: {galleryTitle}">
                        <fui:Gallery label="Gallery" id="imageGallery"/>
                        <mx:ControlBar id="bar">
                                <mx:Label text="{galleryStatus}"/>
                                <mx:Spacer width="10" />
                                <fui:GalleryProgressBar id="progressBar"/>
                                <mx:Spacer width="10" />
                                <mx:Label text="Pages:" />
                                <mx:NumericStepper id="pageStepper"/>
                                <mx:Link id="goButton2"/>
```

```
                                        <mx:Link label="Sebis User-Centered SocialSoftware"/>
                              </mx:ControlBar>
                    </mx:Panel>
          </mx:VDividedBox>

</mx:Panel>
</mx:Application>
```

MXML is an XML 1.0 language. Every MXML file should begin with an XML declaration :`<?xml version="1.0"?>`. The `xmlns` attribute declares an XML namespace. Namespaces mechanism enables using multiple XML tags within a single XML document. Flex uses the namespaces of custom components to locate those components in its classpath. The `:mx` namespace here is a standard one for the MXML class library and is included in every MXML file. The :s refers to the directory where custom classes – Search Components – reside .

The `id` attribute is available for nearly all Flex classes which can be used to provide a unique identifier for an instance of the class. An explicit id value is required unless this object is to be referred in data bindings or ActionScript.

Generally the visual components of a Flex application are placed inside containers, which provide bounding boxes for text, controls, images, and other elements. Here, the Panel container is used to provide the overall visual wrapper. The title attribute of the Panel is automatically included at the top of the panel therefore can be used to display title text in a title bar. In this case they are called "Social Organizer".

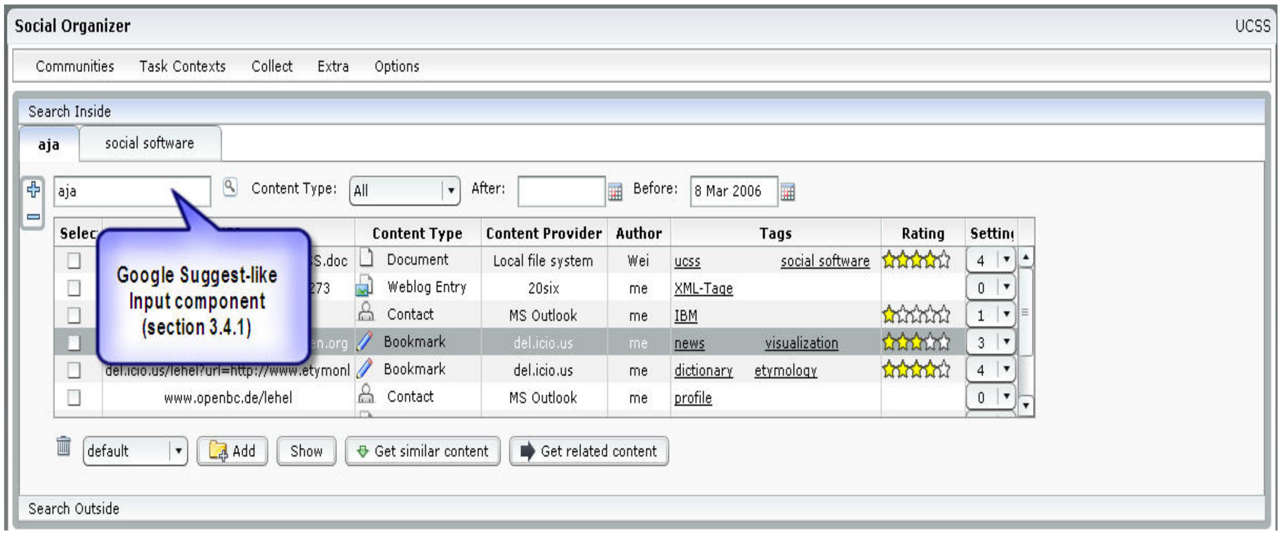
Inside panel there are two blocks divided: MenuBar and VDividedBox. MenuBar displays the menu texts from `menudata.xml` through the faceless component Model. VDividedBox features the resizing effect between its child containers; here are a HBox and a gallery Panel.

A Container called Accordion enables only one of its child containers to be visible at one time. Figure 3.6 show that the current visible element is Search Inside. TabNavigator can also consist of a collection of child containers and help guide the user navigate by using its TabBar. SearchIn is placed as a child of the TabNavigator and it is a reused custom component. Some functions implemented in `searchIn.mxml` like `newTab()` provide to dynamic generation of this component.

The example Photo Gallery is the downside part of the VDividedBox. It contains a custom class named Gallery which will display the thumbnail images resulted from the search query in Search Outside, and a ControlBar which can include a set of visual elements like Label, Link, etc. In figure 3.3 the photos displayed are from Flickr with search keywords "social software".

Another essential part of MXML file is its `<mx:Script/>` tag. Functions and attributes of a Flex class can be declared between this tag or in an external ActionScript file. Latter need to be imported using the `source` attribute of Script.

3.3.1 Search Inside Panel



aja | Content Type: All | After: | Before: 8 Mar 2006

Selec		Content Type	Content Provider	Author	Tags		Rating	Settin
☐	S.doc	Document	Local file system	Wei	ucss	social software	★★★★☆	4
☐	273	Weblog Entry	20six	me	XML-Tage			0
☐		Contact	MS Outlook	me	IBM		★☆☆☆☆	1
☐	n.org	Bookmark	del.icio.us	me	news	visualization	★★★☆☆	3
☐	del.icio.us/lehel?url=http://www.etymonl	Bookmark	del.icio.us	me	dictionary	etymology	★★★★☆	4
☐	www.openbc.de/lehel	Contact	MS Outlook	me	profile			0

default

Figure 3.7: Search inside Panel

Component-based building mechanism

Components are the building blocks of all Flex applications – from navigator containers, to layout containers like HBox, to the controls such as Links introduced in the last section. They define what is seen by the user and how the user can interact with the application.

There is already a broad range of components shipped with Flex, but it's also necessary to create own components sometimes. Flex's component architecture makes it possible to build custom components and provides two different ways to do so: either via pure MXML or through creating an ActionScript 2 class. Which strategy to use is up to the developer and the specific problem, but empirical methods should be taken into consideration before going down a certain path. Creating custom components in MXML is a faster process, but the flexibility of a class-based development environment in which functionality can be encapsulated into multiple classes, increasing reusability across the custom components and testing these components using unit testing frameworks are lost. Therefore MXML-only components have minimal reusability. Outside of the Flex environment brand new components can also be built by Flash MX 2004 and reused by Flex.

Layout description

The Search inside Panel displayed in Figure 3.7 was implemented as class `searchIn.mxml` that mixes the two solutions, creating an MXML component that utilizes ActionScript classes.

Inside `searchIn.mxml` there is another reusable MXML component named `Suggest.mxml`, which performs Google suggest [Goog05] - alike functionalities if user enters keywords in the TextInput control.

Functionalities

Search queries can be collected from keywords of user inputs or take from suggestion list, and the content type selected from a combo box, or the specific date from date component. The results will be shown in the DataGrid component below. The DataGrid has a list of predefined contents available for the user to select from, but they will also be able to add new entries to the DataGrid, which can be persisted to a local shared object. Multiple Content can be selected from the DataGrid, dragged and dropped into trashcan, added to some portfolio ("Add" button), viewed included metadata in a new window ("Show" button). The "Get similar content" button can open the Search outside Panel, dynamically generate a new tab and then get search results there which are contents that are similar to the selected ones. The button "Get related content" will create a new Search Inside tab and show result there. Some effective features can be found in the DataGrid such as some personalized columns with icons or links, checkboxes or images or combo boxes. They are ActionScript classes that can be referenced to the cellRenderer attributes of the corresponding DataGridColumn component. The code blow shows an example. `ComboBoxCellRenderer.as` extends UIComponent and implements the CellRenderer API of Flex.

```
<mx:DataGridColumn columnName="comboData" headerText="Setting"
cellRenderer="{ucss.utils. ComboBoxCellRenderer}">
```

All customized CellRenderer Classes for the application *Social Organizer* reside in the **utils** package, they are: `CheckCellRenderer.as, ComboBoxCellRenderer.as, HyperlinkCell.as, IconsCellRenderer.mxml, LinkCellRenderer.as, RatingCellRenderer.mxml, TileCellRenderer.as.`

Local Shared Objects

Local shared objects are generated by the web browser's flash player and used by Flex to store persistent data on the client machine. The data permitted to store are simple types like Number, String, Boolean, XML, Array or Object, but not functions or Movie Clips. The default capacity is 100kb. Beyond the limit the user will be prompted to allow more data to be stored locally by means of flash player configuration tools. At the initiation of the application, corresponding shared objects can be generated under `Application Data\Macromedia\Flash Player\` directory:

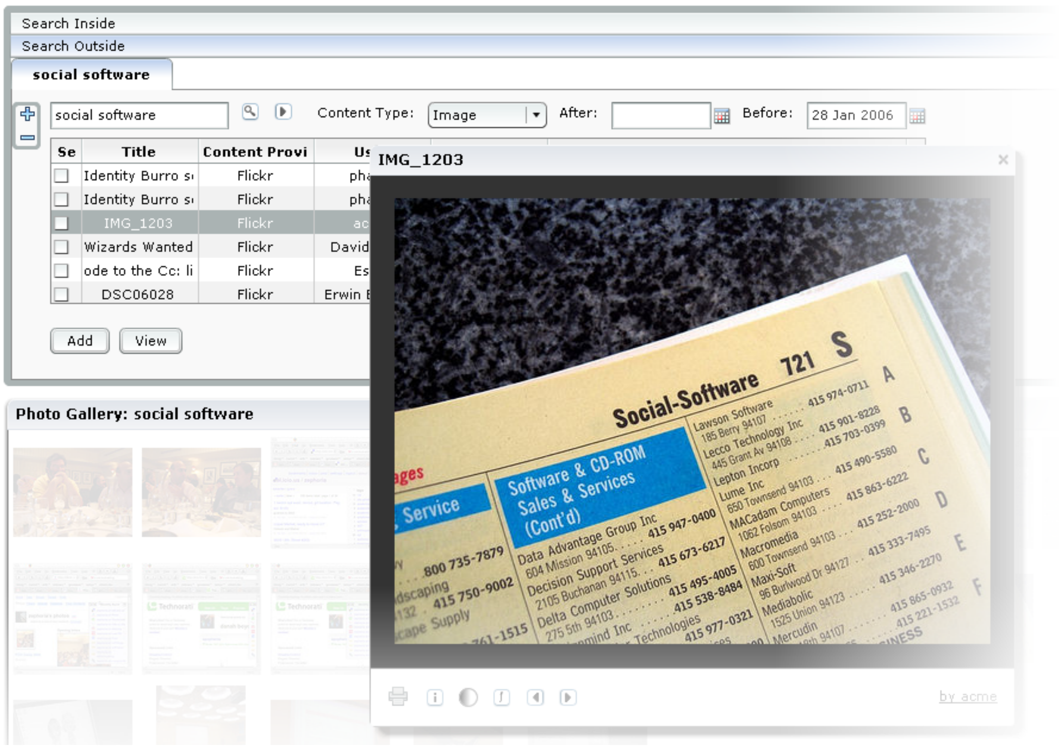
`CommunityManager.sol`	:	object converted from `preCommunities.xml`
`ContentManager.sol`	:	object converted from `preContentsRefs.xml`
`ContentTypeManager.sol`	:	object converted from `preContentTypes.xml`
`ContextManager.sol`	:	object converted from `preCommunities.xml`
`PortfolioManager.sol`	:	object converted from `prePortfolios.xml`
`TagsManager.sol`	:	object converted from `preTagLib.xml`

After that the system accesses the requested data from these shared objects and saves modified data to these shared objects.

RIA Features

Every component has a built-in `toolTip` attribute. Text can be added to explain the function of this component thus the application seems friendly to users.

3.3.2 Search Outside Panel


Search Inside
Search Outside
social software

social software · Content Type: Image · After: · Before: 28 Jan 2006

Se	Title	Content Provi	Us
☐	Identity Burro s	Flickr	pha
☐	Identity Burro s	Flickr	pha
☐	IMG_1203	Flickr	ac
☐	Wizards Wanted	Flickr	David
☐	ode to the Cc: li	Flickr	Es
☐	DSC06028	Flickr	Erwin E

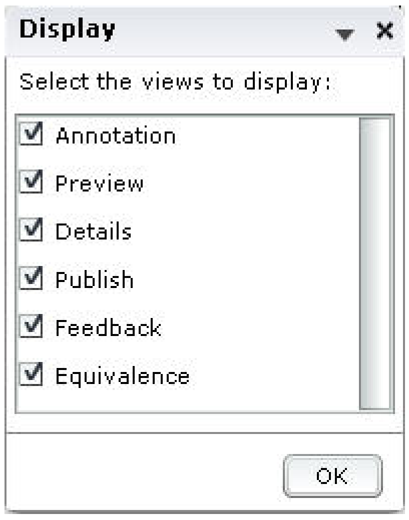
Figure 3.8: Search outside Panel

In search outside panel user can collect contents from online communities through various social software services and display them with metadata in the DataGrid. The columns of the DataGrid are dynamically changeable when the user selects a content type from the combo box above. This means, the *Social Organizer* defines different metadata to display for each content type. Figure 3.8 shows the current search keyword-“social software”. By invoking corresponding Flickr service API all images tagged with “social software” are retrieved and displayed in the DataGrid. The User can double click any item of this DataGrid to view it in a popup window or to add it to the repository of the *Social Organizer*.

3.3.3 Content Panel

A number of tabs offer the opportunity to perform operations on the currently selected content – such as handling annotations, previewing this content, viewing metadata or modifying certain of them, publishing this content on some communities, checking the feedback from others. By default the six tabs are created for all kinds of content, but they can also be configured at the menu: **Options -> Display**

A look of content panel is shown in Figure 3.10. It has been set to display all these tabs for a document file.


Display

Select the views to display:

- ✓ Annotation
- ✓ Preview
- ✓ Details
- ✓ Publish
- ✓ Feedback
- ✓ Equivalence

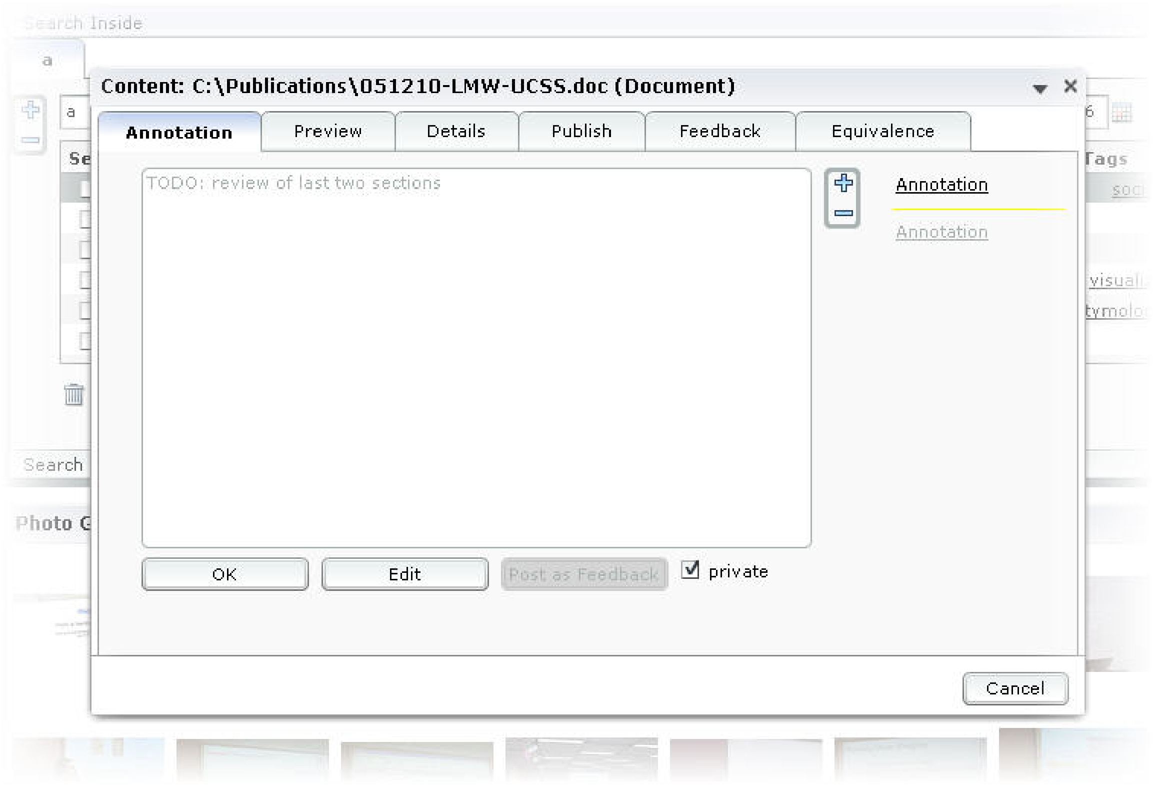
OK

Figure 3.9: Display Panel

3.3.3.1 Annotation Tab

The use cases in the annotation tab are:

Creating a new annotation by clicking the symbol “+”; deleting an annotation by clicking the symbol “-”; an existing annotation can be editable after clicking the “Edit” button, saving current annotation by clicking the “OK” button. Setting an annotation as private results in the invisibility of the “Post as Feedback” button, otherwise an annotation can be also published as feedback.


Content: C:\Publications\051210-LMW-UCSS.doc (Document)

TODO: review of last two sections

Figure 3.10: Annotation Tab

3.3.3.2 Preview Tab

For different content types corresponding preview tabs will be dynamically generated. Figure 3.11 is an example for content type: image. Figure 3.12 is an example for content type: document.

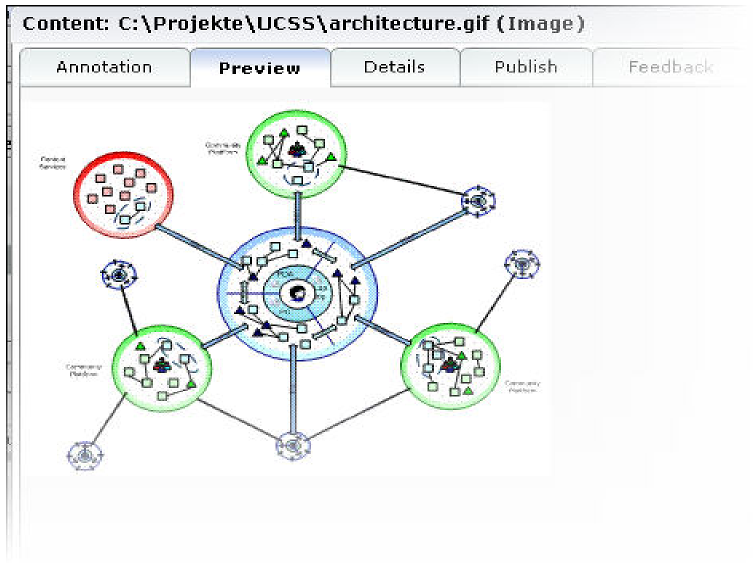
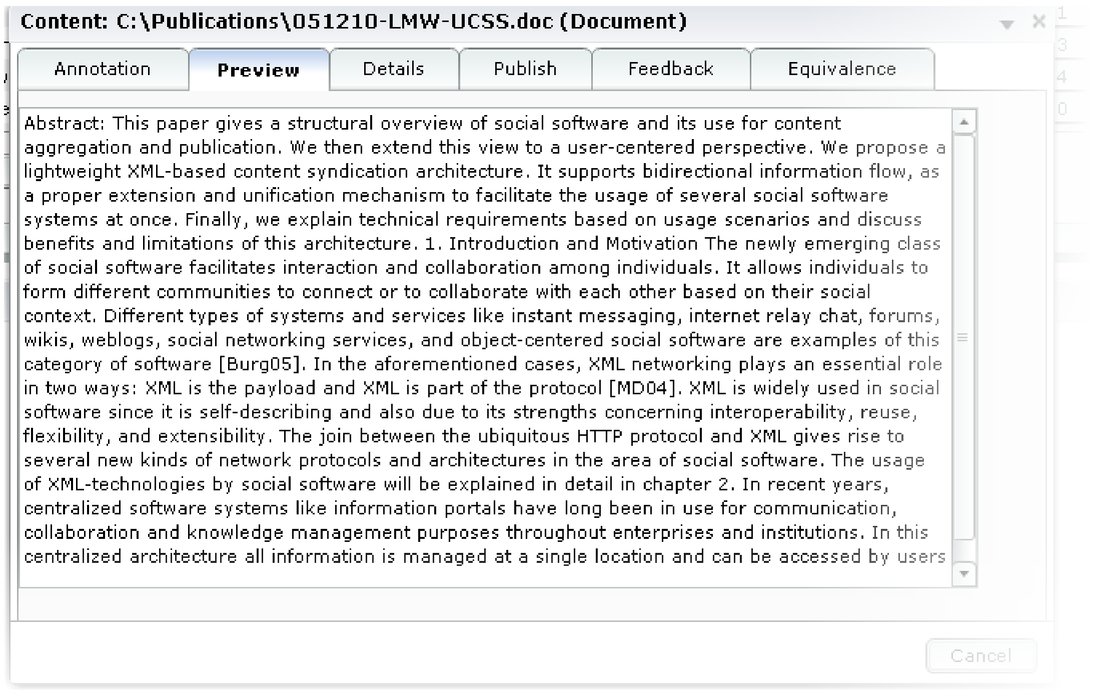


Figure 3.11: Preview Tab for selected Content Type: Image

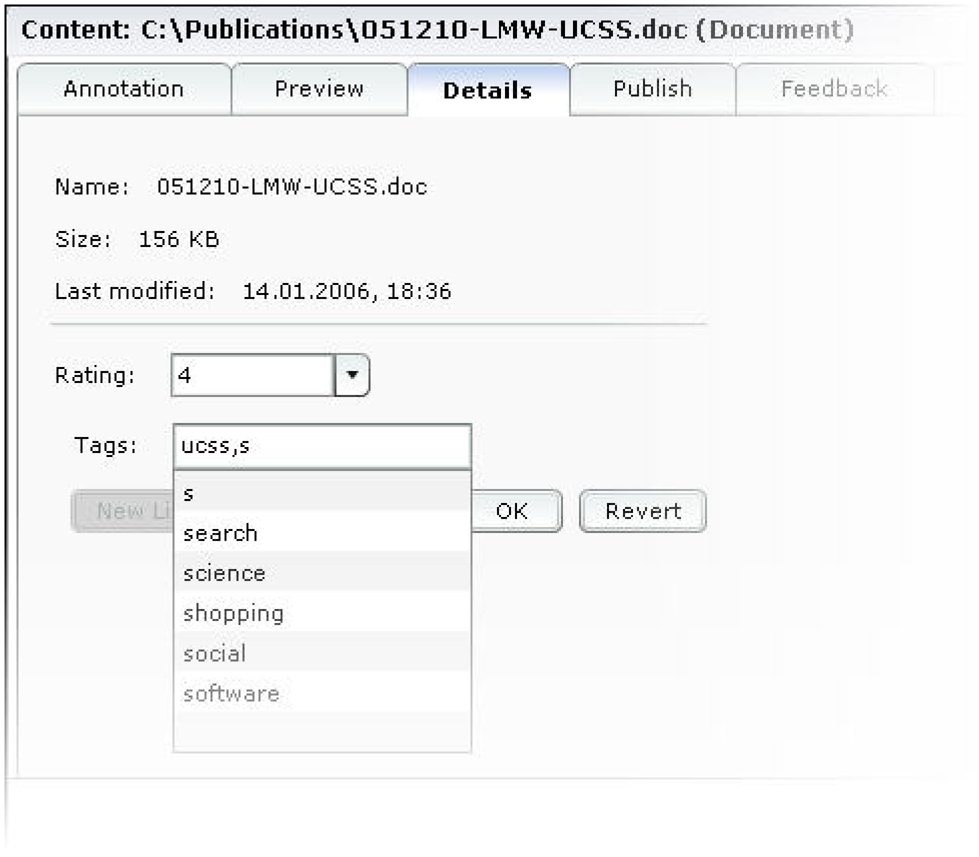


Abstract: This paper gives a structural overview of social software and its use for content aggregation and publication. We then extend this view to a user-centered perspective. We propose a lightweight XML-based content syndication architecture. It supports bidirectional information flow, as a proper extension and unification mechanism to facilitate the usage of several social software systems at once. Finally, we explain technical requirements based on usage scenarios and discuss benefits and limitations of this architecture. 1. Introduction and Motivation The newly emerging class of social software facilitates interaction and collaboration among individuals. It allows individuals to form different communities to connect or to collaborate with each other based on their social context. Different types of systems and services like instant messaging, internet relay chat, forums, wikis, weblogs, social networking services, and object-centered social software are examples of this category of software [Burg05]. In the aforementioned cases, XML networking plays an essential role in two ways: XML is the payload and XML is part of the protocol [MD04]. XML is widely used in social software since it is self-describing and also due to its strengths concerning interoperability, reuse, flexibility, and extensibility. The join between the ubiquitous HTTP protocol and XML gives rise to several new kinds of network protocols and architectures in the area of social software. The usage of XML-technologies by social software will be explained in detail in chapter 2. In recent years, centralized software systems like information portals have long been in use for communication, collaboration and knowledge management purposes throughout enterprises and institutions. In this centralized architecture all information is managed at a single location and can be accessed by users

Figure 3.12: Preview Tab for selected Content Type: Document

3.3.3.3 Details Tab

Details tab shows a list of metadata of the selected content. Some metadata such as rating and tags can also be modified. The tags input area uses the same component `Suggest.mxml` as in search inside panel and search outside panel. The user can scroll up and down to select a description to define this content or insert a new vocabulary. In the future the function of this Suggest component could be better enhanced. For example, if the inserted words are new to the system, they should be remembered and appear in the drop down list the next time.


Content: C:\Publications\051210-LMW-UCSS.doc (Document)

Name: 051210-LMW-UCSS.doc

Size: 156 KB

Last modified: 14.01.2006, 18:36

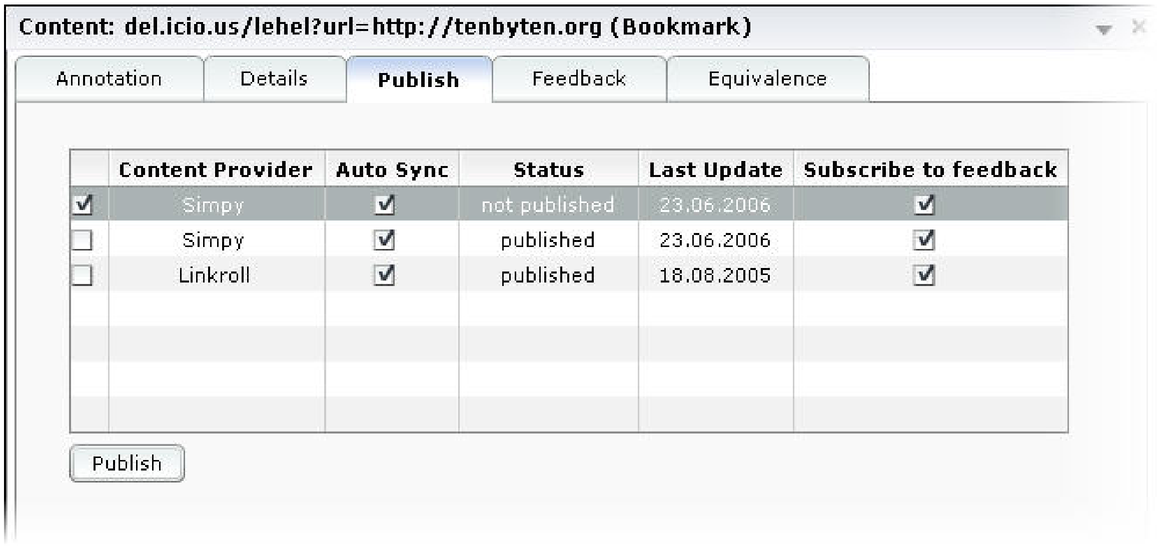
Rating: 4

Tags: ucss,s

Figure 3.13: Details tab

3.3.3.4 Publish Tab

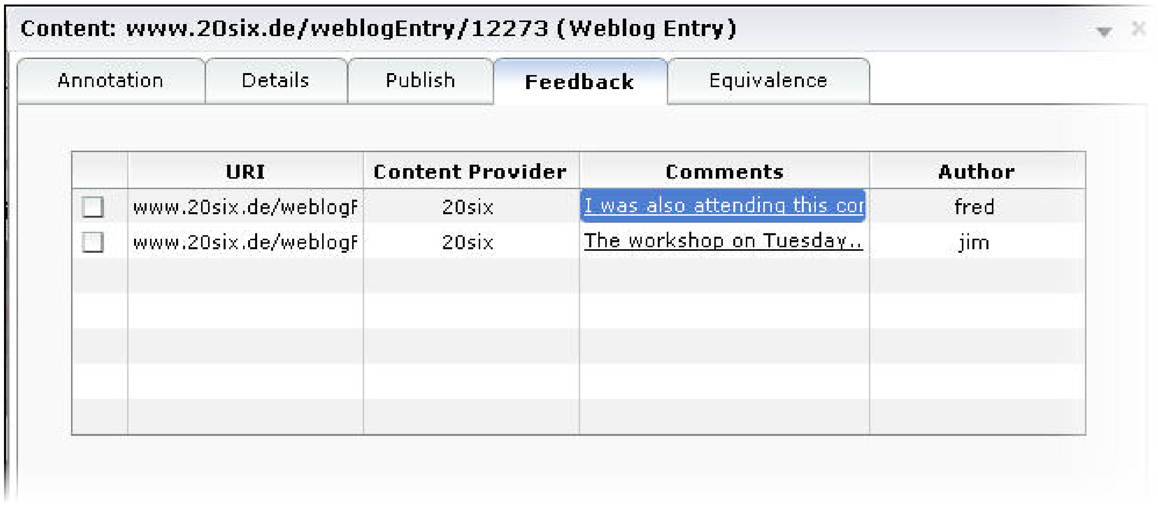
The publish tab displays a DataGrid to show the publishing community platforms and the corresponding information. Figure 3.14 shows the appearance of one publish tab. By clicking the “Publish” button below the selected items can be published to other systems.


Content: del.icio.us/lehel?url=http://tenbyten.org (Bookmark)

	Content Provider	Auto Sync	Status	Last Update	Subscribe to feedback
✓	Simpy	✓	not published	23.06.2006	✓
☐	Simpy	✓	published	23.06.2006	✓
☐	Linkroll	✓	published	18.08.2005	✓

Figure 3.14: Publish Tab

3.3.3.5 Feedback Tab


Content: www.20six.de/weblogEntry/12273 (Weblog Entry)

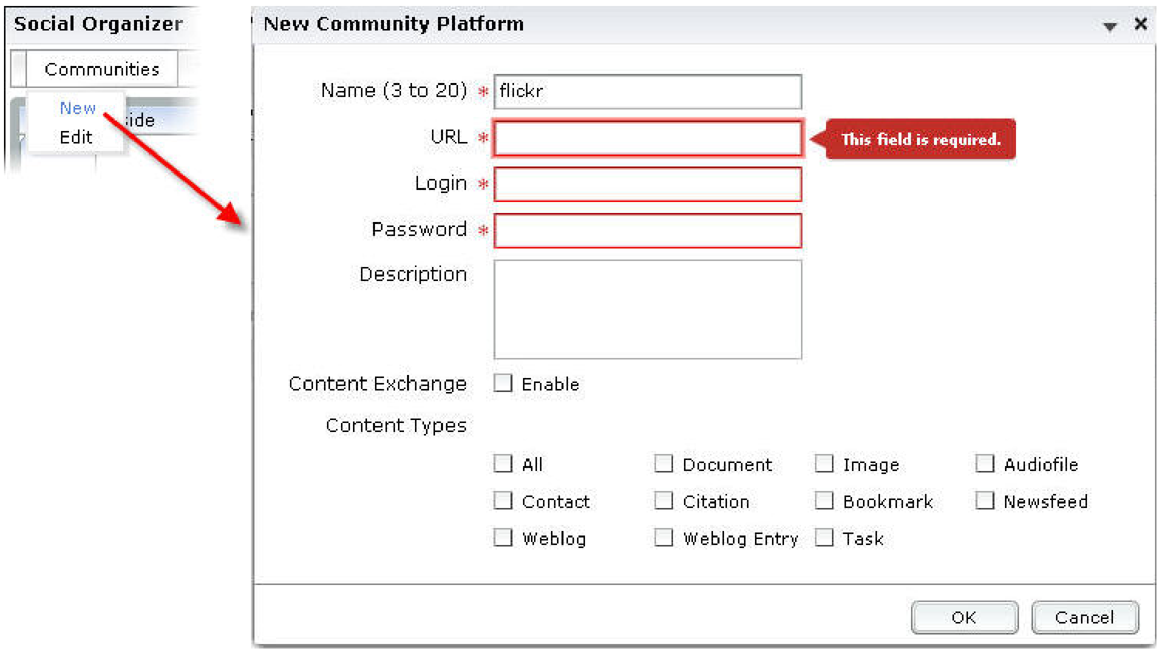
	URI	Content Provider	Comments	Author
☐	www.20six.de/weblogF	20six	I was also attending this cor	fred
☐	www.20six.de/weblogF	20six	The workshop on Tuesday..	jim

Figure 3.15: Feedback Tab of a Weblog Entry

By integrating appropriate services of community platforms in the *Social Organizer*, the user can get feedback automatically. Concerning the user interface for this use case, feedbacks are listed in the DataGrid that contain a preview of the comment text. When clicking the link of comments as shown in Figure 3.15 user can navigate to see the whole comments posted on the related community platforms.

3.3.4 New Community Platform Window

New community platform panel is a popup window comprising a form-based layout. Flex provides a set of built-in validation classes such as StringValidator, DateValidator, CreditCardValidator etc. to simplify the processing of data input detection and data validation. These classes can be found in the package mx.validators of the ActionScript API. Figure 3.16 shows an example using StringValidator and the red marked parts prompting the user that they are not allowed to be left empty before finishing operations. As well as the Flex data validation mechanism, custom validation logic can also be applied to perform this process.



Name (3 to 20) *	flickr
URL *	
Login *	
Password *	
Description	
Content Exchange	☐ Enable

This field is required.

Content Types

☐ All	☐ Document	☐ Image	☐ Audiofile
☐ Contact	☐ Citation	☐ Bookmark	☐ Newsfeed
☐ Weblog	☐ Weblog Entry	☐ Task	

Figure 3.16: New Community Platform Window

3.3.5 Look and Feel of the Application Social Organizer

Styles are useful for defining the look and feel (appearance) of an application. The common “Halo” appearance (see Figure 3.17) of Flex components can be modified individually or unified through style properties. The latter needs to be defined once and then applied it to all. The Flex Style Explore [Expl05] implemented by Flex consulting group is a comprehensive online application for better understanding and operating styles in Flex.

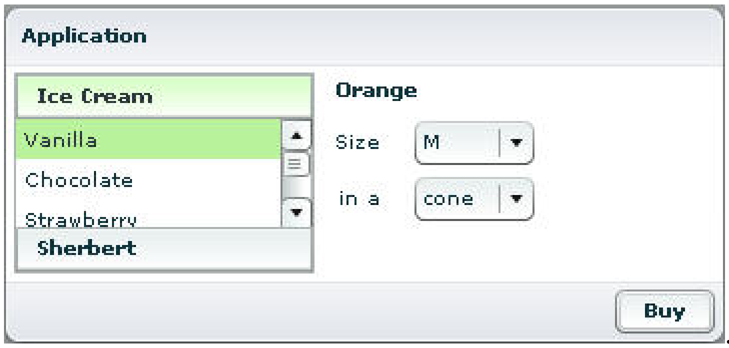


Figure 3.17: A Simple Application with “Halo” Style [Expl05]

CSS

Using Cascading Styles Sheets (CSS) specification to change the style of a document or entire applications is the most convenient means which Flex supports, but least flexible. Style Sheets can define global styles that are inherited by all controls, or individual classes of styles for certain controls. The following example applies the external style sheet `mac_osx.css` to the application *Social Organizer*. The Macintosh style can be explored from previous introduction of the user interface.

```
<mx:Style source="ui_css/mac_osx.css"/>
```

Flex includes a global style sheet defined in the flex-config.xml that is the basis of all style definitions applied to all applications. By default, this global style sheet has no style definitions, but it provides a convenient location to define them.

Themes and Skins

Furthermore, themes and skins can also be applied to the style of an application conferring more flexibility. Using the theme property of the `<mx:Application>` tag a single theme can be applied to the Flex application. The following example applies the `mac_osx.swc` theme to the application *Social Organizer*:

```
<mx:Application xmlns:mx="http://www.macromedia.com/2003/mxml" theme="
ui_css/mac_osx.swc ">
```

3.3.6 Deployment of the Application Social Organizer

HTML Wrapper

At sending a MXML request to the browser, the default behavior is for Flex to return an HTML wrapper pointing to a generated SWF file. In this case the generated SWF for the *Social Organizer* is `index.mxml.swf` which can be found in the source code of the HTML file. This HTML wrapper includes supports for history management and could be used as a template for creating a customized wrapper in the future.

Precompiling SWF files

Precompiling SWF files from MXML source code can be done by using the mxmlc compiler or requesting them in a browser and then saving the generated files. Precompiled SWF file can be simply added later to any existing dynamic or static web page by adding the HTML wrapper needed.

Requirements of the Deployment

The platform used in this developing process is Microsoft Windows XP Professional. The Java version used is JDK 1.5.0. The IDE applied here is Flex Builder. The software chosen here is Flex 1.5. Flex is a J2EE web application that can be installed as a Web Application Archive (WAR) file to any supported Java application server. Flex includes the following WAR files:

- flex.war — The primary Flex WAR file
- samples.war — Sample Flex applications
- profiler.war — Debugging application for use with Flex

Since I select to install Flex to an integrated Java Application Server (JRun 4.0), it is needed is to deploy these Flex WAR files to **JRun4/servers/default** directory before start building Flex applications. Similarly, if using Tomcat as the application server, these Flex WAR files should be deployed to **Tomcat/webapps** directory. Besides, Flash Player 7.0.14 or later should be installed in the system as well.

To view the application *Social Organizer*, run the following link in the browser (if using JRun):

http://localhost:8700/flex/SocialOrganizer/index.mxml

4 Conclusion and Outlook

This chapter summarizes the experiences and draws some conclusions in order to make better use of RIA technologies to improve the features of the Social Organizer.

4.1 Conclusions

Rich Internet Applications can address the challenge which traditional web application couldn't achieve and therefore draw a promising future for the improved user experience upon their internet activities. This benefit has been shown by more and more successful RIAs emerging on the web. There are already several RIA solutions available currently as choices depending on the specific problem.

This report covers the various RIA solutions, analyzes their features and shows their advantages as well as disadvantages in different aspects. The references presented are purposed to contribute in building and architecting Rich Internet Applications.

Furthermore, as best practice Adobe Flex 1.5 was chosen to develop the user interface of the *Social Organizer* – a user-centered social software as an extension of personal information management. From the user interface introduced in section 3 it is seen that Flex 1.5 benefits the implementation of the *Social Organizer* UI. Even though some issues like inconvenienced debugging and code hunting exist in Flex 1.5, they are solved in Flex 2.0. From the developers' point of view, using Flex is an efficient way to create applications with rich user experiences.

4.2 Outlook

The above considerations present the starting point. Along with the announcement of Flex 2.0 some new considerations need to be taken into account. Comparing with formal versions the new generation seems to be more matured and attractive in a number of aspects such as: abundant build-in UI components, rich collection of class libraries, enterprise services, development environment, programming language, and conjunction with Ajax technologies. The *Social Organizer* can step by step adopt these new advantages of Flex 2.0. Some already existing frameworks like Cairngorm framework [Cair05] - microarchitecture (collection of design patterns) for RIAs [Webs06], FAST framework providing application services for logging and tracing, and value-adding class libraries extending the Flex Framework's own implementation of RPC data services in Flex 1.*x* [FAST05], and Flex Unit Testing Framework [Unit05] greatly improve the *Social Organizer* and other large scalable RIAs.

5 Reference Materials

[Alla02] Jeremy Allaire: Macromedia Flash MX – A next generation rich client, http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf, Access: June 2005.

[Bect04] Becta: Thin client networking http://www.becta.org.uk/subsections/foi/documents/technology_and_education_research/thin_client.pdf#2, Access: October 2005.

[Cair05] Cairngorm framework, http://www.iterationtwo.com/open_source_cairngorm.html, Access: December 2005.

[Coen06] Christophe Coenraets: Publish/Subscribe Messaging and End-to-End Persistence for Rich Internet Applications, http://labs.macromedia.com/wiki/index.php/Flex_Enterprise_Services:overview, Access: January 2006.

[CMSW05] Christophe Coenraets (Macromedia), Vincent Mendicino (SAP), Natalia Shmoilova (SAP), Dirk Wodtke (SAP): Creating Next Generation SAP Analytics Applications with SAP NetWeaver and Macromedia Flex, http://www.macromedia.com/software/flex/whitepapers/pdf/sap_flex.pdf, Access: February 2005.

[Croc01] Douglas Crockford: JavaScript:
The World's Most Misunderstood Programming Language, http://www.crockford.com/javascript/javascript.html, Access: August 2005.

[Deli05] Del.icio.us, http://del.icio.us, Access: May 2005.

[Dome06] Marc Domenig: Rich Internet Applications and AJAX - Selecting the best product, http://www.javalobby.org/articles/ajax-ria-overview/, Access: January 2006.

[Duhl03] Joshua Duhl: Rich Internet Applications, http://download.macromedia.com/pub/solutions/downloads/business/idc_impact_of_rias.pdf, Access: April 2005.

[Expl05] Flex Style Explorer http://weblogs.macromedia.com/mc/archives/FlexStyleExplorer.html, Access: October 2005.

[FAST05] Flex Application Starter Toolkit, http://www.macromedia.com/devnet/flex/articles/fast_userguide.html, Access: December 2005.

[FBui06] Flex Builder 2.0, http://labs.macromedia.com/wiki/index.php/Flex_Builder:overview, Access: January 2006.

[FEnt06] Flex Enterprise Services, http://labs.macromedia.com/wiki/index.php/Flex_Enterprise_Services, Access: January 2006.

[Flex04] Press Releases, http://www.macromedia.com/macromedia/proom/pr/2004/flex_available.h tml, Access: October 2005.

[Flex05] Press Releases, http://www.macromedia.com/macromedia/proom/pr/2005/announcing_fle x2.html, Access: October 2005.

[Flic05] Flickr, http://www.flickr.com/, Access: September 2005.

[Forr06] Forrester Research, http://www.forrester.com/my/1,,1-0,FF.html, Access: January 2006.

[Gmai05] Gmail, http://www.gmail.com/, Access: February 2005.

[Gmap05] Google Maps, http://maps.google.com/, Access: February 2005.

[Gsug05] Google Suggest, http://www.google.com/webhp?complete=1&hl=en, Access: October 2005.

[Garr05] Jesse James Garrett: Ajax: A New Approach to Web Applications, http://www.adaptivepath.com/publications/essays/archives/000385.php, Access: October 2005.

[Humm06] Hummingbird Ltd, http://www.hummingbird.com/index.html?cks=y, Access: January 2006.

[Iter05] iteration:: two, http://www.iterationtwo.com/open_source_cairngorm.html, Access: October 2005.

[Kay01] Michael Kay: What kind of language is XSLT? http://www-128.ibm.com/developerworks/xml/library/x-xslt/?article=xr, Access: October 2005.

[Lehe05] Vanda Lehel: User-Centered Social Software – Beyond Closed Community Platforms (slides), Chair sebis in TUM, 2005

[Lehe06] Vanda Lehel: User-Centered Social Software - Model and Characteristics of a Software Family for Personal Information Management (slides), Chair sebis in TUM, 2006

[LMW05] Vanda Lehel, Florian Matthes, Sheng Wei: User-Centered Social Software – Beyond Closed Community Platforms, Chair sebis in TUM, June 2005.

[Link06] Linkedin, http://linkedin.com, Access: January 2006.

[Long06] Longhorn, http://msdn.microsoft.com/longhorn, Access: January 2006.

[Macr03] Macromedia whitepaper: Developing Rich Internet Applications with Macromedia MX 2004, http://www.macromedia.com/devnet/studio/whitepapers/rich_internet_apps.pdf, Access: October 2005.

[McLe05] Drew McLellan: Very Dynamic Web Interfaces, http://www.xml.com/pub/a/2005/02/09/xml-http-request.html, Access: September 2005.

[Open06] OpenBC, https://www.openbc.com/, Access: January 2006.

[Regu06] Regular Expression, http://labs.macromedia.com/svn/flashplatform/?/projects/actionscriptsamples/tags/0_1_0/src/actionscript3/regularexpressions/RegularExpressions/, Access: January 2006.

[Rour04] Cameron O'Rourke: A Look at Rich Internet Applications, http://www.oracle.com/technology/oramag/oracle/04-jul/o44dev_trends.html, Access: November 2005.

[Unit05] Flex open source Unit Testing Framework, http://www.iterationtwo.com/open_source_flexunit.html, Access: December 2005.

[Wadh05] David Wadhwani: Flex 2.0: Enabling the Next Generation of Rich Internet Applications, http://www.macromedia.com/devnet/flex/articles/flex2_intro.html, Access: December 2005.

[Webs06] Steven Webster: Developing Flex RIAs with Cairngorm Microarchitecture, http://www.macromedia.com/devnet/flex/articles/cairngorm_pt1.html, Access: January 2006.

[Wei05] Coach K. Wei: AJAX: Asynchronous Java + XML? http://www.developer.com/design/article.php/3526681, Access: December 2005.

[Wiki05] Rich Internet Application, http://en.wikipedia.org/wiki/Rich_Internet_Application, Access: September 2005.

[XmlU05] Open Source XML UI Toolkits in Java, http://java-source.net/open-source/xml-user-interface-toolkits, Access: December 2005.

[Ymap05] Yahoo Maps, http://maps.yahoo.com/, Access: August 2005.