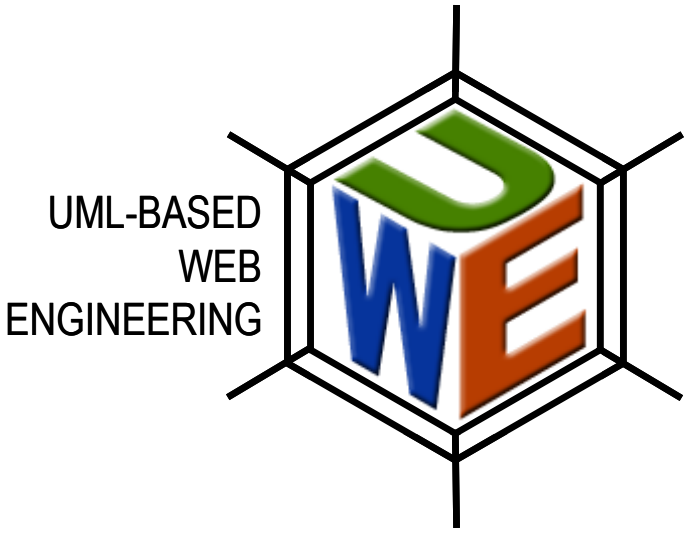

Rich Internet Applications

State-of-the-Art

Marianne Busch and Nora Koch

Technical Report 0902
Programming and Software Engineering Unit (PST)
Institute for Informatics
Ludwig-Maximilians-Universität München, Germany
December 2009

Rich Internet Applications – State-of-the-Art
Version 1.0 – December 2009
Technical Report 0902

Marianne Busch and Nora Koch

Ludwig-Maximilians-Universität München (LMU), Germany
Institute for Informatics
Programming and Software Engineering (PST)
uwe.pst.ifi.lmu.de

This research has been partially supported by the project MAEWA II "Model-based Engineering of adaptive Rich Internet Applications" (WI841/7-2) of the Deutsche Forschungsgemeinschaft (DFG), Germany and the EC 6th Framework project SENSORIA "Software Engineering for Service-Oriented Overlay Computers" (IST 016004).

EXECUTIVE SUMMARY

A new kind of web applications came up to overcome the limitations of traditional web applications regarding the usability and interactivity of their user interfaces: the so-called Rich Internet Applications (RIAs). New languages and frameworks mainly drive the development of RIAs, but engineering of RIAs is a very new field in the area of software engineering in which a precise definition of RIAs and specification of their characteristics is still missing.
This report presents a discussion on RIAs definitions, characteristics and the state-of-the-art regarding engineering approaches for RIAs and technologies used in their implementation. Based on existing definitions of RIAs this report came up with a new definition that addresses both, the technical characteristics of RIAs, and the user experience.

Rich Internet Applications (RIAs) are web applications, which use data that can be processed both by the server and the client. Furthermore, the data exchange takes place in an asynchronous way so that the client stays responsive while continuously recalculating or updating parts of the user interface.
On the client, RIAs provide a similar look-and-feel as desktop applications and the word "rich" means particularly the difference to the earlier generation of web applications. RIAs are basically characterized by a variety of interactive operating controls, the possibility of on-/offline use of the application, and the transparent usage of the client and server computing power and of the network connection.

TABLE OF CONTENTS

1 Introduction

Traditional web applications have great limitations regarding the usability and interactivity of their user interfaces. A new kind of web applications came up to overcome these limitations: the so-called Rich Internet Applications (RIAs). This type of applications offers a series of advantages, e.g. they provide a richer and more efficient graphical interface resembling desktop applications. RIAs are complex web applications based on thick client architecture, asynchronous communication, and a great variety of user interface widgets. On the one hand, the client will manage data and processes; on the other hand, network traffic is reduced by the improved communication. Both innovations allow for additional user interface features increasing the usability and the interaction possibilities of the users.

The development of RIAs, which are more complex software than web applications, requires also a more complex development process. Requirements engineering has to cope with the variety of features that RIAs can offer to the users. The design of RIAs has to consider different type of architectures and communication and desktop-like user interfaces. Implementation of RIAs is based on the integration of specific libraries that support the different user interface features.

Engineering of RIAs is a very new field in the area of software engineering in which a precise definition of RIAs and specification of their characteristics is still missing. New languages and frameworks mainly drive the development of RIAs where the main problem is the lack of documentation. Recently, a set of modelling and model-driven approaches have been proposed. Some of them have been designed as extension of engineering approaches defined for the development of web applications, such as OOHRIA (extension of OO-H) [3], UWE-R [8], WebML [14] [17], and OOHDM extension [6].

This report presents the state-of-the-art regarding engineering approaches for RIAs and technologies used in the implementation of RIAs, such as different Ajax and JavaScript frameworks. Beforehand we present a discussion on RIAs definitions and characteristics. A categorisation of RIAs is proposed as well and future trends are mentioned at the end of this report.

2 What are Rich Internet Applications?

The aim of this section is to discuss several definitions of RIAs in order to form our own well-founded opinion about it. Currently, a great diversity of definitions is available on the Web, but most of them only compare RIAs to desktop applications in a very general way suffering on lack of accuracy.

Wikipedia for example defines:

Rich Internet applications (RIAs) are web applications that have some of the characteristics of desktop applications, typically delivered by way of a proprietary web browser plug-ins or independently via sandboxes or virtual machines. Examples of RIA frameworks include Adobe Flex/AIR, Java/JavaFX, uniPaaS and Microsoft Silverlight.

The term was introduced in March 2002 by vendors like Macromedia who were addressing limitations at the time in the "richness of the application interfaces, media and content, and the overall sophistication of the solutions" [12] by introducing proprietary extensions. [10]

Examples for "*characteristics of desktop applications*" are drag and drop, the use of keyboard shortcuts, the utilization of the local computing power and immediate feedback to users when they interact with the application.

Two similar definitions are those written by Santiago Meliá et al. [3] and Paul and Harvey Deitel [13]:

Traditionally, Web applications have had great limitations in the usability and interactivity of their user interfaces. To overcome these limitations, a new type of Web applications called Rich Internet Applications (RIAs) has recently appeared providing richer and more efficient graphical components similar to desktop applications.

RIAs are web applications that offer the responsiveness, "rich" features and functionality approaching that of desktop applications. Early Internet applications supported only a basic HTML graphical user interface (GUI). Though they could serve simple functions, these applications did not have the look or feel of a desktop application. [..]RIAs are a result of today's more advanced technologies that allow greater responsiveness and advanced GUIs. (p. 32)

We agree with these definitions, but wondered, if we cannot describe what RIAs are in a more precise way. On the one hand, RIAs are certainly characterized by their different ways of utilisation compared to traditional web applications. On the other hand, the look-and-feel of RIAs is based on a couple of technologies, which are working together.

This definition of Bozzon et al. [14] reinforces the role that plays the client-server architecture and describes the usual workflow.

[RIAs] 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 RIA 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.

The aspect of "*minimizing client-server data transfers"* is capable of being misunderstood. Although it is true that for instance some compressed raw data can be finally calculated on the client-side, the overall data volume is not minimized, but usually divided into several requests, each one transferring only the actually needed information. Consequently, the principal point is asynchronism, as the user no longer has to reload the whole website for each data stream, as stressed in the definition of Leonardo Machado et al. [8]:

RIA are applications that benefit from the web ubiquitousness and enhance the traditional web application model adding rich user interface elements altogether with a flexible and asynchronous server interaction mechanism. The final user experience resembles that of a desktop application, since the user is no longer required to wait for a synchronous server response for each link that is activated.

Another definition claims "*RIAs generally split the processing across the Internet/network divide by locating the user interface and related activity and capability on the client side, and the data manipulation and operation on the application server side*" [16]. We think that this definition is not general enough, as data and process distribution can differ from RIA to RIA and rather depends on the task the application is designed for. Besides, we want to avoid vague statements like a "*RIA normally runs inside a Web browser and usually does not require software installation on the client side to work. However, some RIAs may only work properly with one or more specific browsers.*" [16].

In addition, it can be confusing to enumerate several concrete frameworks, as seen in the Wikipedia article above, because in this field, the technologies change quickly and RIAs should not be connected only with a subset out of them. (For technologies that are used for the implementation of RIAs, see chapter 5.)

To conclude, we come up with the following definition that we think expresses best the difference between traditional web applications and this new kind of software for the web called Rich Internet Applications (RIAs). The first section deals with the technical part of RIAs, the second with the particular experience the user may have.

Rich Internet Applications (RIAs) are web applications, which use data that can be processed both by the server and the client. Furthermore, the data exchange takes place in an asynchronous way so that the client stays responsive while continuously recalculating or updating parts of the user interface.
On the client, RIAs provide a similar look-and-feel as desktop applications and the word "rich" means particularly the difference to the earlier generation of web applications. RIAs are basically characterized by a variety of interactive operating controls, the possibility of on-/offline use of the application, and the transparent usage of the client and server computing power and of the network connection.

In our opinion, a "web application" is not only defined by its interaction with "*a network such as the Internet or an intranet*" [11], but rather by the focus on its navigation structure. Wikipedia claims a web application has to be "*accessed via a web browser*" [11], but we think this restriction may become more and more obsolete, as the difference between web desktop application is vanishing (see chapter 6: "Future Trends").

3 RIAs Distinguishing Characteristics

Rich Internet Applications are more interactive and more responsive applications than traditional web applications. The basic characteristics are **unnecessary page reload, drag&drop facilities, short response time and multimedia animations**. Based on these characteristics, different functionalities such as live validation, auto completion, periodic refresh, and even rich text editors can be offered to the RIA user.

The distinguishing characteristics of RIAs are related to technological features used to build this kind of web applications. These features are **data distribution**, **distribution of page computation**, **asynchronous communication** between client and server and **enhanced user interface behaviour**.

In the following subsections, we go into further details of these different technological features, which make the distinction from traditional web applications possible. The resulting

RIA architectures correspond to so-called fat clients. Conversely, so-called thin clients characterize traditional web applications. The developer has to decide which software architectural elements are required to support the additional functionality and storage capacity of the fat client [9], such as browser-side cache, predictive fetch and multi-stage download.

3.1 Data Distribution

In traditional web applications, data resides on the server. In RIA applications data can be distributed between both, server and client. The developer can decide about the distribution and even design an application that may temporarily be used irrespective of the server. Therefore, a RIA can use the client's persistent and volatile content. Data can be manipulated on the client, and finally sent to the server once the operation has been completed [17].

The advantages of the distribution of data on client and server side are:
→ offline usage
→ validation and preparation of data on client side

The disadvantages of the distribution of data are well-known problems of the database field:
→ data replication
→ policies for allocation
→ consistency of data

3.2 Distribution of Page Computation

In traditional web applications, there is only one controller at the server side, which orchestrates the computation of the page. At each user interaction, the whole page is computed by scratch and reloaded. In RIAs a second controller at client side is introduced that is responsible for the computation and refreshment of a portion of a page [17]. Data processing can be executed both at client and server side. RIAs have a different navigation structure from Web 1.0 applications. Due to the augmented process capability of the client, in RIAs both the client and the server can carry out complex operations [21].

The advantages of a distributed page computation are:
→ live validation
→ offline usage
→ page rearrangement [9]
→ display morphing [9]

The disadvantages of a distributed page computation are:
→ client requires additional data

3.3 Client-Server Communication

In RIAs mechanisms to minimize data transfers are introduced which move interaction and presentation layers from the server to the client. Conversely to web applications, RIAs use both synchronous and asynchronous communications. Pull and push communication capabilities are available. Distribution of data and functionality across client and server broadens the features of the produced events as they can be originated, detected, notified, and processed in a variety of ways [21].

The advantages of synchronous and asynchronous communication are:
→ partial page refreshment
→ page rearrangement [9]
→ display morphing [9]

The disadvantages of a more complex client-server communication:
→ incremented development efforts

3.4 Enhanced User Interface Behaviour

RIAs offer enhanced interface behaviours and improved user interactions (e.g. multimedia support, animations and drag&drop).

The main advantages of the enhanced user interface behaviour are:
→ operation as single page applications
→ avoid unnecessary refreshments of the whole page
→ allow progressive presentation loading when needed [21]

The disadvantages of more sophisticated interface behaviour:
→ performance problems
→ browser incompatibilities

4 Modelling Approaches for RIAs

In a model-driven or model-based approach, we need a model language that provides model elements that allow us to model these distinguishing features of RIAs. Depending on the level of abstraction, we require different type of model elements. On business level (computational independent model in the MDA terminology), we need model elements, which allow a draft representation of the features that the RIA will provide to the user, i.e. the facilities the user will perceive. On design level these features need a refinement indicating e.g. more precisely how is the user interaction and the client-server communication. Finally, when building the platform specific model or at code level, the technologies used need to be specified in detail.

The need of an engineering support for RIA development has been recently addressed by several methods. We distinguish four types of approaches:

1. The extension of an already existing method with RIA features for modelling, additional transformations, and specific generation mechanisms, such as OOHRIA, OOHDM, UWE-R and WebML.
2. Combination of a method for the development of web applications with a method for user interfaces design and generation, e.g. RUX combined with WebML and UWE.
3. A new method for designing and implementing RIAs, e.g. the ADRIA approach and the work of Martinez-Ruiz et al.
4. Pattern-based approach as the design recommendations based on best practices, e.g. library proposed by Scott or Mahemoff and the UML-based patterns proposed by UWE.

4.1 OOH4RIA

OOH4RIA extends the OOH method introducing many new model elements for modelling two additional models: the presentation and the orchestration models of RIAs, which complement the OO-H models for domain and the navigation of a RIA. The presentation model captures the different widgets used for the user interface. The orchestration model represents the interaction between the widgets and the rest of the system. In [3] a model-driven approach proposes the use model transformations to implement RIAs.

The implementation is based on the Google Web Toolkit (GWT), which is an AJAX framework. The approach creates Java code for the server-side application, and HTML and JavaScript for the client-side code. The OOHRIA models similarly to OO-H models are specified using UML by means of MOF meta-models.

4.2 OOHDM Extension

In Urbieta et al. [6] issues related to behaviour, single-page paradigm and content composition are treated. This work conceptually extends the method OOHDM for modelling RIAs supporting a very abstract specification of the RIA user interface. Platform specific generation is not considered.

4.3 UWE-R

UWE-R [8] is a light-weighted extension of UWE for RIAs, covering navigation, process and presentation aspects. Hence, new modelling elements are defined that inherit structure and behaviour from UWE elements. In contrast to our work, UWE-R uses stereotypes for many of the extensions instead of meta-attributes.

4.4 WebML Extension

Regarding extensions of existing methods, it is worth to mention the work of Toffetti et al. that propose in [14] the modelling of distributed data and events in data-intensive RIAs focusing on client or server side actions. WebML distinguishes between data distributed on client-side and server-site as well as persistent and temporary objects. In particular, WebML is extended by enriching data specifications with two dimensions: (1) data location and (2) data duration. Location can be either server or client; duration can be persistent or temporary.

WebML introduces new modelling elements for the hypertext model for the computation on client-side. They show how events can be explicitly described and coupled to the other concepts of a web modelling language in order to specify collaborative RIAs. WebML is extended by model elements to indicate complete computation of a content unit, content refreshment without predetermining the values of the input parameters, invalidation of the values of the content and/or the input parameters. For more details on the WebML extension for RIAs the interested reader is referred to [14].

4.5 OOWS and Interaction Patterns

The OOWS method proposes to create an interaction model to address RIA features [22]. In order to define the interaction model, the interaction pattern concept is used to describe a solution for a common user-system interaction. These interactions patterns are presented as guidelines for producing the RIA interface code based on a set of transformation rules. Such an interaction pattern models the structural aspects of the pattern; the behavioural aspects are only described textually. No details are given on the transformation rules in [22].

4.6 RUX Method

The RUX method combines modelling of the presentational aspects of RIAs with an existing method for designing web applications, such as in [4] where UWE is complemented with the RUX method for the UI design. The approach consists of the transformation of UWE presentation model to a RUX abstract interface model (AIM), which is afterwards enriched

with typical RIA user interface actions. In this approach, RIA features are introduced into models at a lower level of abstraction than in the current approach.

In [4] we propose to combine UWE and RUX methods to generate RIAs. The approach consists of the transformation of the UWE presentation model to a RUX abstract interface model (AIM), which is afterwards enriched with typical RIA user interface actions. In a second step the AIM is transformed to a concrete interface model taken into account among others device specific capabilities. In this approach, RIA features are introduced into models at a lower level of abstraction than in other approaches, such as OOHRIA and UWE-R.

Preciado et al. [4] and Brambilla et al. [21] follow the second type of approach combining RUX and WebML. They propose to model a mix of conceptual models for addressing design and development of web applications supported by rich interfaces in [4]. The business modelling language (BPMN) is used for describing workflows, which are then translated to a WebML specification of a web application implemented according to the Single Page Paradigm, typical of RIAs. Finally, the RUX-Method introduces features for refining the rich interface design.

4.7 ADRIA Approach

Dolog and Stage [18] also proposes a new method called ADRIA, which employs interaction spaces, tasks models and state machines. This method focuses on the design of events triggered by user interactions. Disadvantage of such a new method is that in case of reengineering of existing web applications it requires modelling from scratch.

4.8 Model-driven Approach proposed by Martinez-Ruiz et al.

The work of Martinez-Ruiz et al. [19] proposes a new mode-driven method for designing graphical user interfaces in RIAs decomposing the presentation design into several abstraction levels. The method is based on XSL model transformations and the focus is on the implementation techniques.

4.9 Pattern Libraries

The Scott pattern library is based on the work from the internal pattern library (Matt Leacock, Erin Malone, Kiersten Lammerding, Chanel Wheeler and others) with a number of new patterns created to express the rich interaction of Ajax style applications [15].

Patterns in the book of Mahemoff [9] are just a concise way to represent the knowledge embodied in the many AJAX applications. The point is to discover best practices by investigating how developers have successfully traded off conflicting design principles. AJAX is all about usability, so the patterns focus particularly on delivering usability in the face of constraints, most notably: user capabilities and expectations, bandwidth limits, the stateless nature of HTTP, the complexity of JavaScript.

4.10 UWE Patterns

Patterns have proved valuable for efficient RIA programming [9]. UWE propose to apply patterns at a higher abstraction level, i.e. modelling, to achieve the objectives of minimizing the design efforts and maximizing the expressiveness of the models used in the development of RIAs. Our focus is on the use of state machines for the representation of the patterns – a widely used modelling technique. UWE patterns are therefore used to describe the behaviour

of the RIA features; in contrast to the interaction patterns introduced in [22] that only model the structure of the pattern. The models of the RIA patterns we specify can be embedded in almost all existing methodologies. In this sense, it is a general approach for all UML conform methods. The use of these RIA patterns only requires the definition of extension points in the methodology, and afterwards the specification of how to integrate them in a hosting language such as UWE, which makes them easily reusable [4].

5 Technologies for RIA's Implementation

Our definition of RIAs (see chapter 2) does not refer to any specific technology for the implementation of a RIA. In this chapter, we establish categories that can be used for a classification of a concrete RIA. Ideally, these aspects are considered while designing and implementing the software. In addition, existing RIA frameworks and their features are discussed.

5.1 Categories of RIAs

Not every RIA only uses "native", scripting based web technologies (e.g. JavaScript, HTML and XML) as AJAX applications do. Many depend on browser plugins like Flash[1] or even require software like Java Web Start[2], which has to be installed on the client separately.

For interface definition languages XUL[3] and XAML[4], Toffetti [17] introduces an extra class called "browser-based". We do not want to adopt this, because XUL and XAML are not solely relevant in the context of RIAs without considering the associated programming language or rather the associated framework (e.g. Microsoft Silverlight[5]).

One problem is that plugins and script languages can be deactivated or not even be installed, which makes the web application unusable in most cases. Often, a general solution can only contain plain HTML that means no *rich* application is possible at all. Nowadays it is unusual that a new browser comes without JavaScript or (Flash) plugin support, but in many companies the browsers are not up-to-date. Furthermore, due to security issues, features that enable RIAs can be turned off.

If the RIA can be executed, the next question is, if you can go offline while using it. For example, Gears[6] is a general approach that synchronizes the offline data later.

Other points are the requirements to the computing power and hardware of the client and of the corresponding server (e.g. the client's video card).

The kind of navigation structure has to be considered, because it is an integral part of any web application, as mentioned in chapter 2. Some applications, like simple Flash games may not include a dedicated navigation structure and we got to the point, that those examples are more related to media objects, like embedded videos, than to web applications. For a web engineer this differentiation leads to touch choices, which last but not least depend on the fact whether the navigation structure is worth to be modelled or not.

[1] Adobe Flash Player. http://www.adobe.com/de/products/flashplayer/
[2] Java Web Start. http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp
[3] Mozilla XUL. https://developer.mozilla.org/en/XUL
[4] Microsoft XAML. http://msdn.microsoft.com/en-us/library/ms747122.aspx
[5] Microsoft Silverlight. http://www.microsoft.com/silverlight/
[6] Gears. http://gears.google.com/

Further categories are applicable for desktop applications as well, but especially the security aspect is important. On one hand, there is the risk of bad encryption and eavesdropping of data transmissions, because communication issues are an essential part of web applications. On the other hand, the browser, the runtime environment, and the web application should take care of the client's safety.

The following list provides a summary of categories; a concrete RIA can be classified with:

- *Used runtime environment*, such as scripting-based, plugin-based and web-based desktop technology.
- *Distribution of the RIA*, such as only-online and offline.
- *Resources of the client and the server*, such as computationally intensive, video card intensive and memory intensive.
- Kind of *navigation structure*.
- *Safety & security issues*, such as sandbox behaviour and data encryption.

This classification may be supported by the general features of RIAs mentioned in chapter 3. For example, it can be useful to estimate the percentage of page computation on the client in order to compare several implementations of similar applications with regard to the available resources. In certain circumstances, the evaluation of the quality of user interaction and usability may be in the spotlight. Anyway, each RIA has to face the difficulty of various requirements; therefore our categories have to be adapted and prioritized in particular cases, e.g. to support a selection process.

5.2 Features of RIA frameworks

There exists many frameworks and it is very difficult to give an overview. Nevertheless, we want to mention some features of the available RIA frameworks. Table 1 contains Ajax frameworks; Table 2 includes frameworks that need a dedicated runtime environment. The latter are called "container frameworks" because the applications at least control a predefined part of the web page and are dependent on a container like a browser plugin. Another approach is to install such a "container framework" locally, e.g. to execute RIAs outside of the browser. Currently, no clear border has been established between *plugin-based* and *web-based desktop technologies*. For example, an application built with Silverlight 3 can be executed within the browser just as well as separately.

We have selected only a few Ajax frameworks out of a large number of existing ones, each one covering slightly different aspect (see Table 1).

Name	Language	Other characteristics
Prototype JavaScript framework[7]	JavaScript	JavaScript library. Used in e.g. in Ruby on Rails.
script.aculo.us[8]	JavaScript	Extends the Prototype JavaScript framework.
MooTools[9]	JavaScript	Based on the Prototype JavaScript framework.
jQuery[10]	JavaScript	May in the future be included in Microsoft's Visual Studio (ASP.net Ajax and ASP.net MVC Framework) and in Nokia's Web Run-Time platform.

[7] Prototype JavaScript framework. http://prototypejs.org/
[8] script.aculo.us. http://script.aculo.us/
[9] MooTools. http://mootools.net/
[10] jQuery. http://jquery.com/

Name	Language	Other characteristics
Yahoo! UI Library (YUI)[11]	JavaScript	Includes GUI controls, e.g. a Rich Text Editor and provides a large set of utilities.
Ext JS[12]	JavaScript	Originally an extension of the Yahoo! UI Library. Provides several widgets.
Dojo Toolkit[13]	JavaScript	Includes a packaging system and widgets.
Google Web Toolkit[14]	Java, JavaScript	JavaScript Native Interface (JSNI) allows writing parts of JavaScript within the Java code.

Table 1: Ajax frameworks

It can be difficult to choose the most convenient framework for a particular project. The decision may be based on the familiarity with one of the frameworks as well as on the popularity and of course, the functional range. The fact of being open source also plays an important role.

The familiarity with programming languages and techniques may also be decisive, e.g. Java: Google Web Toolkit, Java Server Faces: Richfaces[15], C++: Wt[16], .NET: Ajax.NET[17] or ASP.NET AJAX[18] etc. Beginners might become an overview using Wikipedia's List of Ajax frameworks[19] and the Comparison of JavaScript frameworks[20] or reading e.g. [18], which covers Prototype, jQuery, YUI, ExtJS, Dojo and MooTools.

Name	Developer	Language	Runtime Environ-ment	Open Source	Other characteristics
Java Applets[21]	Sun Micro-systems	Java	Java Runtime Environment (jre)	Partly	Normally runs within a browser. Compared to Java Web Start (which executes java applications) applets have access to the actual browser context.
JavaFX[22]	Sun Micro-systems	JavaFX Script	JavaFX Runtime (comes with jre)	Partly	Can be used on the desktop, within a browser and on mobile phones.
Adobe Flash[23]	Adobe Systems	ActionScript	Flash Player Plugin	except[24]	Focus on audio, video and animations.

[11] YUI – The Yahoo! User Interface Library. http://developer.yahoo.com/yui/
[12] Ext JS. http://www.extjs.com/
[13] The Dojo Toolkit. http://dojotoolkit.org/
[14] Google Web Toolkit. http://code.google.com/intl/en/webtoolkit/
[15] RichFaces Project Page – Jboss Community. http://www.jboss.org/richfaces
[16] Wt, C++ Web Toolkit. http://www.webtoolkit.eu/wt
[17] Ajax.NET. http://www.ajaxpro.info/
[18] ASP.NET AJAX http://www.asp.net/ajax/
[19] Wikipedia: List of Ajax frameworks. http://en.wikipedia.org/wiki/List_of_Ajax_frameworks
[20] Wikipedia: Comparison of JavaScript frameworks. http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks
[21] Java Applets. http://java.sun.com/applets/
[22] JavaFX. http://www.sun.com/software/javafx/
[23] Adobe Flash. http://www.adobe.com/de/products/flash/
[24] Open Source Flash http://osflash.org

Name	Developer	Language	Runtime Environ-ment	Open Source	Other characteristics
Adobe Flex[25]	Adobe Systems	MXML, ActionScript	Flash Player Plugin	-	Runtime environment "Flex builder" (based on eclipse) supports drag & drop to create GUIs.
Adobe Integrated Runtime (AIR)[26]	Adobe Systems	Flash, Flex, HTML and Ajax	Adobe Integrated Runtime	-	Offline access (out of browser). Supports also PDF files.
Microsoft Silverlight (Linux: Moonlight[27]**)**	Microsoft	.NET languages; Scripting languages; XAML	Silverlight Plugin	Partly (Moon-light)	Offline access. Includes a Media player. Similar to the Windows Presentation Foundation (WPF[28]).
OpenLaszlo[29]	Laszlo Systems	LZX, JavaScript	Flash, JavaScript	√	Actually Open Source. Similar to Adobe Flex.
UniPaaS[30]	Magic Software	UniPaaS	UniPaaS	-	Allows to change between Full Client and RIA application. Can include .NET components.
Curl[31]	Curl Inc.	Curl	Curl Surge RTE	-	Sandbox for offline applications. Development environment "Curl Surge Lab IDE"
Omnis[32]	TigerLogic Corp	Omnis	Omnis web client	-	Integrated development environment called "Omnis Studio"

Table 2: RIA "container frameworks"

The choice of a "container framework" poses a similar challenge to the software engineer. As mentioned in the former subsection, the average spread of the chosen runtime environment among the potential users additionally plays an important role. For example, a RIA used within an Intranet of a company is more likely to encounter compatible browsers and plugins. Further distinguishing aspects are the price of the framework and the market success of the software producer.

Even if a wide range of RIA frameworks exists, nearly all claim to support recent browsers and provide e.g. drag and drop, rich text editors, auto completion tools, event handling, visual effects and user interface widgets. The difficulty is the usage of buzzwords as Ajax, Web 2.0, RIAs and "offline access" in descriptions (e.g. Wikipedia[33]), which leads to the situation that it is unavoidable to test some frameworks before making a decision for a concrete project.

[25] Adobe Flex. http://www.adobe.com/de/products/flex/
[26] Adobe AIR. http://www.adobe.com/go/air
[27] Moonlight. http://www.mono-project.com/Moonlight
[28] WPF is a part of the Microsoft .NET framework
[29] OpenLaszlo. http://www.openlaszlo.org/
[30] uniPaaS. http://www.magicsoftware.com/2559-en/uniPaaS.aspx
[31] Curl. http://www.curl.com/
[32] Omnis Studio. http://www.omnis.net/index.html
[33] Wikipedia: Comparison of web application frameworks. http://en.wikipedia.org/wiki/List_of_web_application_frameworks

6 Future Trends

Finally, we want to point out the main trends regarding the leading edge technologies. Our RIAs definition in chapter 2 states: "*(RIAs) are Web applications*" and we primarily thought of applications that are used within a browser. Currently, the focus is moving from browser dependent applications to the web applications themselves. Mozilla Prism[34] is one example of reducing the difference to desktop applications by handling them in the same way. Gears also includes a desktop module and provides on/offline transparency with a synchronisation mechanism when going online again. Another example is the Google Chrome Operating System,[35] which is thought to handle web applications especially user-friendly.

This trend to expand RIAs in a way that they play the same role as desktop applications (or even a kind of desktop application, s. Adobe AIR) will lead to different requirements for the engineering process. On one hand, there will be fewer reasons for writing a desktop-only application. On the other hand, the modelling techniques supporting RIAs will have to be enhanced in order to cope not only with the special features of RIAs, but also with the engineering process of complex software. Thus, approaches based on successful software engineering methods seem to be promising. This was one of the main reasons to design UWE as a lightweight extension of the UML. For example, typical navigation issues of web applications can be modelled conveniently, which is difficult with plain UML.

To sum up, it is assumed that the rising importance of RIAs makes high demands on the future software engineers. For this reason, we think the web engineering approaches for RIAs have to be strengthened continuously in order to meet the growing demand and to cover the increasing application range.

7 Conclusion

We presented a concise state-of-the-art of the engineering aspects of RIAs starting with an overview on several definitions of Rich Internet Applications and presenting a more explicit one. We describe the main characteristics of this kind of web applications, which comprise data distribution, distribution of page computation, client-server communication and enhanced user interface behaviour. These aspects require distinguishing model elements for building RIA models and specific features that have to be provided by the technologies used for the implementation of RIAs.

We discussed the currently existing engineering approaches, in particular WebML, OOHRIA, OOHDM extension, RUX approach, OOWS, UWE-R, etc. classifying them into extensions of existing methods, combination of methods, new and pattern-based approaches.

We categorized RIAs based on the runtime environment they use, the offline facilities of the application, the resources used on client and server side, and the safety and security issues. In addition, a set of frameworks for the implementation of RIAs are analyzed and compared. Ajax frameworks are shown in a separate table from, the so-called "container frameworks". Some criteria for the selection of an appropriate framework are discussed as well. The success of RIAs depends on the tool support for both specification and implementation. Anyway, it will be exciting to keep track of future trends.

[34] Mozilla Labs - Prism. http://labs.mozilla.com/prism/
[35] Chromium OS (The Chromium Projects). http://www.chromium.org/chromium-os

References

[1] Nora Koch, Alexander Knapp, Gefei Zhang and Hubert Baumeister. UML-based Web Engineering: An Approach based on Standards (book chapter). In Web Engineering: Modelling and Implementing Web Applications. Gustavo Rossi, Oscar Pastor, Daniel Schwabe and Luis Olsina (Eds.), Springer, HCI, 2008

[2] Nora Koch, Matthias Pigerl, Gefei Zhang, and Tatiana Morozova. Patterns for the Model-Based Development of RIAs. In Proc. 9th Int. Conf. Web Engineering (ICWE 2009), volume 5648 of LNCS, pages 283-291. Springer Verlag, 2009.

[3] Santiago Meliá, Jaime Gómez, Sandy Pérez, Oscar Díaz. A Model-Driven Development for GWT-based Rich Internet Applications with OOH4RIA. Proc. 8th Int. Conf. on Web Engineering (ICWE'08). IEEE, pp.13 – 23, 2008.

[4] Juan Carlos Preciado, Marino Linaje, Rober Morales, Fernando Sánchez-Figueroa, Gefei Zhang, Christian Kroiß and Nora Koch. Designing Rich Internet Applications Combining UWE and RUX-Method. Proc. of 8th Int. Conf. on Web Engineering (ICWE'08), IEEE, New York, USA, 148-154, 2008.

[5] Giovanni Toffetti, Sara Comai, Allessandro Bozzon, Piero Fraternali. Modeling Distributed Events in Data-Intensive Rich Internet Applications, Proc. 7th Int. Conf. on Web Information Systems Engineering, LNCS 4607, pp. 593-602, Springer, 2007.

[6] Matias Urbieta, Gustavo Rossi, Jeronimo Ginzburg, Daniel Schwabe. Designing the Interface of Rich Internet Applications. Proc. 5th Latin American Web Congress (LA-Web'07), pp.144-153, IEEE, 2007.

[7] Yahoo – Design Pattern Library and UI Library, http://developer.yahoo.com/ypatterns/, http://developer.yahoo.com/yui/, last visited 10.02.2009.

[8] Leonardo Machado, Orlando Filho, João Ribeiro. 2009. UWE-R: An Extension to a Web Engineering Methodology for Rich Internet Applications. *WSEAS Trans. Info. Sci. and App.* 6, 4 (Apr. 2009), 601-610.

[9] Michael Mahemoff. Ajax Design Patterns, O'Reilly, 2006.

[10] Wikipedia. Rich Internet Application. http://en.wikipedia.org/wiki/Rich_Internet_application, 2009

[11] Wikipedia. Web application. http://en.wikipedia.org/wiki/Web_application, 2009

[12] Jeremy Allaire. Macromedia March 2002 requirements for Rich Internet Applications. http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf, March 2002

[13] Paul J. Deitel and Harvey M. Deitel. AJAX, Rich Internet Applications, and Web Development for Programmers, Prentice Hall International, 2008

[14] Alessandro Bozzon, Sara Comai, Piero Fraternali and Giovanni T. Carughi. 2006. Capturing RIA concepts in a web modeling language. In Proceedings of the 15th international Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM, New York, 907-908

[15] Bill Scott. RIA Patterns. Best Practices for Common Patterns of Rich Interaction http://www.uxmatters.com/mt/archives/2007/03/, Last visited 10-02-2009.

[16] What is Rich Internet Application (RIA)? A definition from Whatis.com. http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci1273937,00.html

[17] Giovanni Toffetti. Conceptual Modeling and Code Generation of Data-Intensive Rich Internet Applications. PhD Thesis, page 16, 2007.

[18] Peter Dolog, Jan Stage. Designing Interaction Spaces for Rich Internet Applications with UML. Proc. 7th Int. Conf. on Web Engineering (ICWE'07). LNCS 4607, Springer-Verlag, pp.358-363, 2007.

[19] Francisco J. Martínez-Ruiz, Jaime Muñoz Arteaga, Jean Vanderdonckt, Juan M. González-Calleros, Ricardo Mendoza. A First Draft of a Model-driven Method for Designing Graphical User Interfaces of Rich Internet Applications, In Proc. of the 4th Latin American Web Congress (LA-Web'06), pp. 32-38, IEEE, 2006.

[20] Leslie Michael Orchard, Ara Pehlivanian and Jonathan Snook. Professional JavaScript Frameworks: Prototype, jQuery, YUI, ExtJS, Dojo and MooTools. Wiley & Sons, 2009

[21] Marco Brambilla, Juan Carlos Preciado, Marino Linaje, Fernando Sanchez-Figueroa. Business Process -based Conceptual Design of Rich Internet Applications. Proc. of 8th Int. Conf. on Web Engineering (ICWE'08), IEEE, New York, USA, 155-161, 2008.

[22] Francisco Valverde, Oscar Pastor. Applying Ineraction Patterns: Towards a Model-Driven Approach for Rich Internet Applications Development. Proc. 7th Int. Workshop. on Web-Oriented Software technologies (IWWOST 2008)