The Practical Aspects of Rich Internet Application Development and Quality Factors: RIA–based Decision Support System

Wieslaw Pietruszkiewicz[1] and Dorota Dzega[2]

[1] West Pomeranian University of Technology, Faculty of Computer Science and Information Technology
ul. Zolnierska 49, 71-210 Szczecin, Poland
wpietruszkiewicz@wi.zut.edu.pl
[2] West Pomeranian Business School, Faculty of Economics and Information Technology
ul. Zolnierska 53, 71-210 Szczecin, Poland
ddzega@zpsb.szczecin.pl

Abstract. In this article we present the practical aspects of RIA (Rich Internet Application) development and selected quality factors. In the first part of this paper we discus this branch of internet software. Later, analysing users feedback and focusing on software usability, we propose the major quality factors particular to RIA. Evolving these features we compare pros and cons for the most popular technologies. In the next part of article we show the practical aspects of development internet applications by presenting a web–based Decision Support System called TeamCreator, build to support team members selection. The process of development of this application was performed in a manner supporting its quality. The final part of this article contains an analysis of particular risk sources concerned with the development of advanced internet applications and the common pitfalls observed for many projects.

Key words: Web applications, Rich Internet Applications, Software design, Decision Support Systems

1 Introduction to Rich Internet Applications

The first and the simplest solution to build a web–based systems is a "thin client" architecture, where the client layer is responsible only for the presentation of data and all operations are performed by the server side. This approach significantly reduces quality of web service and according to [1] the Web has been transformed into a platform, where this ecosystem has interoperability capabilities and separates application running environment from operating system or hardware.

The developed RIA (Rich Internet Application) technologies offer better quality (at least in usability part). While [2] distinguishes three major advantages of RIAs i.e. processing users interactions by client layer, asynchronous

communication with server and minimization of page reloads, we also think that RIAs important advantages over desktop applications are:

- simplicity of installation or even often no installation steps,
- safety for users, because the service is less vulnerable to threats (malware),
- availability for users on any machine connected to Internet,
- independency from the hardware platform or operating system,
- easiness of updates and changes that are immediately available for all users.

However, before deciding to build an application in form of RIA their disadvantages must be pointed out and understood:

- requirements for additional software (plug–ins/RIA frameworks),
- lower speed than desktop applications, because code is interpreted and converted into a native code on client machine,
- security restrictions, forbidding access to critical resources,
- rest fully on server and the quality of network connection.

Nowadays, building web services that use pure HTML sacrifices ergonomics for the lower costs and simplicity. Therefore, we decided that a form of Rich Internet Application will be the most appropriate for the developed multi–user Decision Support System (DSS in abbr.) called TeamCreator. This application was created to recommend team members for research projects and was developed in a way ensuring the basic RIA usability factors (discussed later).

2 Major RIA quality factors and popular RIA technologies

Analysing the common pitfalls of RIAs development and feedback from users (during the project described later and the other projects), we gathered a few major quality factors for RIA technologies, that should offer:

- high speed and a fast start–up time,
- multi–platform support,
- no requirements for any additional plug–ins,
- Rapid Application Development (RAD later) tools,
- easy customisation of GUI components,
- GUI attractiveness,
- visual similarity to desktop applications,
- prototyping capabilities,
- flexibility.

These factor focus on usability, being a part of the software quality (including also reliability, efficiency or security), but in our opinion increased usability is a feature distinguishing RIAs. As there are several RIA technologies [3] available

Table 1. The comparison of the most popular RIA technologies

Technology	**Pros & cons**	**Vendor**
AJAX	⊕ does not require additional plug–ins, very popular ⊖ large number of incompatible libraries, the low speed of development, depends on JavaScript engine with varying behaviour on different browsers	-
Flex [3]	⊕ based on Flash available on most computers ⊖ close solution, no available advanced RAD tools – apart the one from vendor	Adobe
OpenLaszlo [4]	⊕: open source solution, high speed of prototyping, ability to compile code to Flash and AJAX, based on Flash or JavaScript available on most computers ⊖: no advanced RAD tools	LaszloSystems
JavaFX [5]	⊕ free RAD tools, uses mature and very popular Java VM ⊖ new and still uncommon technology	Sun
Flash [6]	⊕ mature, popular ⊖ closed solution, technology is oriented on graphics not programming, hard prototyping	Adobe
Silverlight [7]	⊕ uses tools and languages supported by. NET ⊖ closed solution, requires special plug–in not available on different platforms	Microsoft
Java [8]	⊕ well documented solution, high potential, popular ⊖ requires JVM, difficult prototyping	Sun
HTML	⊕ no requirements for additional plug–ins, client machine low load ⊖ low speed of application development, slow, large amount of transmitted data, high server load, low ergonomics	-

on market, we analysed them to later choose one technology, the most suiting the requirements for a RIA–based Decision Supporting System (see Tab. 1).

After technological evaluation of possible solutions, we have selected OpenLaszlo (OL later) as the best meeting our requirements. Originally this technology was introduced as Laszlo Presentation Server, but in 2004 it was released as Open Source and its renamed to OpenLaszlo. The following OpenLaszlo features influenced our choice:

- is available on various Operating Systems and hardware platforms,
- was build with support for application prototyping,
- is visually attractive,

[3] While HTML is not a RIA technology, it was a reference point as a basic, pre–RIA solution.

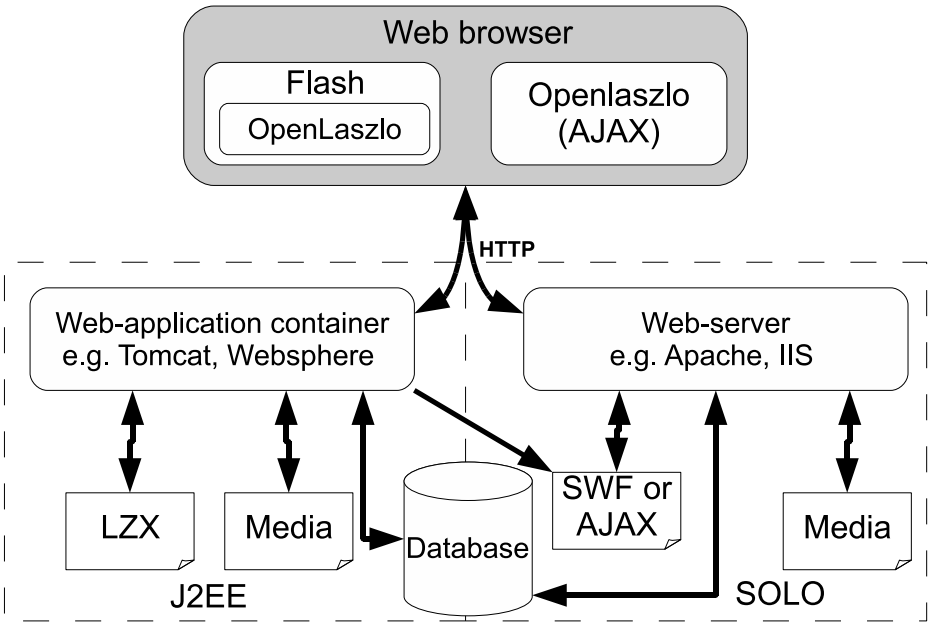

Fig. 1. A simplified OpenLaszlo architecture

- has wide capabilities,
- offers many GUI components,
- rarely does need to install additional software on client machine, as OL compiles into Flash or JavaScript,
- is an open source solution – for a scientific project it was highly possible that some components had to be tailored to custom demands.

The OpenLaszlo architecture was presented on Fig. 1. This technology provides two ways to supply RIA service i.e. "J2EE deployment" and "SOLO deployment" [9]. The first one requires a web applications container (Tomcat in OpenLaszlo installation packs) on production server. The queries relating to the OpenLaszlo application are being processed by the J2EE server. In this case, it is possible to incorporate all features provided by the J2EE server e.g. SOAP service. In the second variant i.e. "SOLO deployment" OL application is compiled by the developer to a Flash clip or AJAX site and later placed on an ordinary HTTP server. Obviously this variant is simpler to use on most of Web servers, but it is impossible to use special J2EE features within.

From a programmer's perspective, the design of OpenLaszlo applications is based on LZX language, being a combination of XML similiar to XUL/XAML (relating to the GUI) languages and ECMAScript – JavaScript/ActionScript (to create program code). It is important that LZX language allows quick creation of application skeleton. However, a significant drawback of this technology is a lack of advanced RAD tools (comparing to Java RADs), that would support the

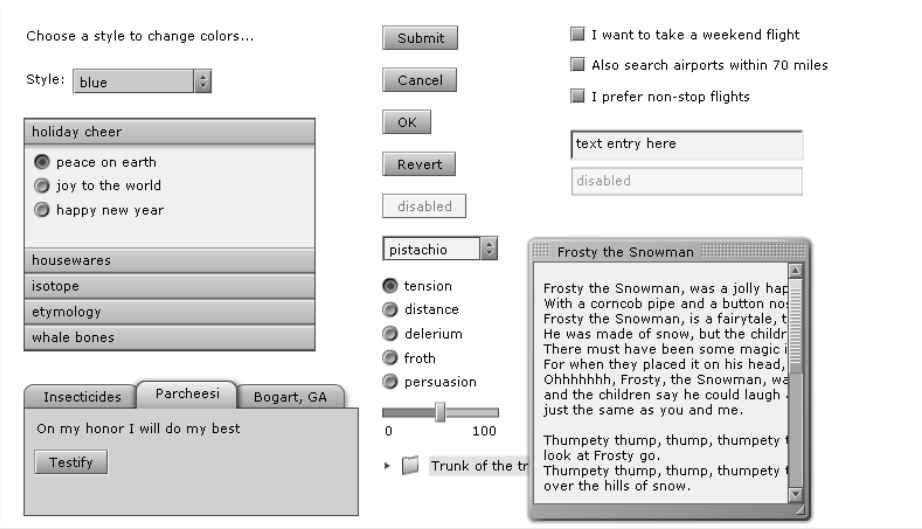

Fig. 2. OpenLaszlo components example – a part of OpenLaszlo Technology Demos

developer after skeleton construction. The OL possibilities are enhanced by large number of GUI components (see Fig. 2). Programming in LZX can by supported by any XML tool, but there exist OpenLaszlo plug–ins to NetBeans and Eclipse.

3 Development of RIA–based Decision Support System

During a research project over a method of team selection for scientific projects we have used OpenLaszlo to build a Decision Support System (DSS in abbr.) – TeamCreator. This application is a multi–user web–based system and its core, being a team efficiency model and optimizing algorithm, came from [10].

From the technical point of view, TeamCreator is a RIA running on two servers and client browser equipped with Flash plug–in. The first server is HTTP server (Apache was used) and the second one is a database server (MySQL was used). However, it is possible to use other serves e.g. IIS server as HTTP server and PostgresSQL as database server. The deployment diagram was presented in Fig. 3.

The client part consists is "`index.php`" file with an embedded OpenLaszlo application compiled to Flash "`team.swf`". These files are used as GUI, while client asynchronously communicates with "`interface.php`" file that serves as an interface to server engine. The core of the system is in "`system.php`" file that uses Pear DB as an interface to communicate with database server.

The database structure was presented on Fig. 4. The tables were designed to store data necessary to ensure the basic systems functionality:

– performing user tests such as personality profile test, team role, team culture and ambition test,

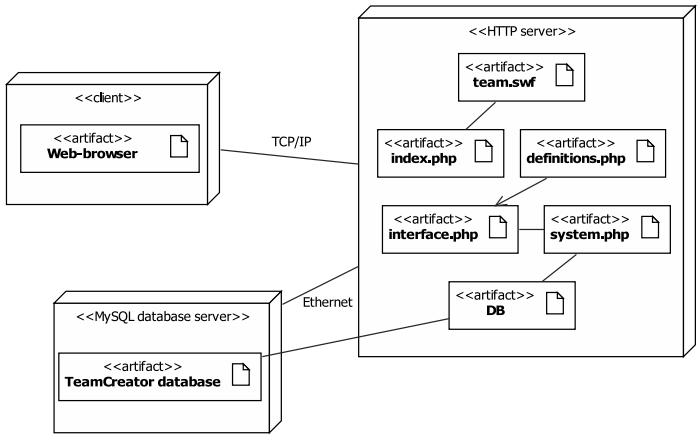

Fig. 3. The deployment diagram for TeamCreator

- gathering information about users skills,
- gathering information about projects requirements,
- perform teams control,
- selection of team members,
- administration functions,
- reports generation.

The whole application was divided onto eight modules (Fig. 5), each was responsible for handling a logically coherent use–cases. The split onto modules was important for the application development, giving a chance to break it into smaller units, which improved the process of creating, testing and introduced a better code organization.

The important step in each software project is a selection of proper life cycle model. For this software, being a part of wider research project, the selected solution had to meet the key requirements:

- high flexibility,
- low effort of software modification,
- reduced excessive planning for functionality requiring prior research,
- analysis of users' feedback,
- frequent verification of the software functionality.

It was also essential, that the quality factors of developed system will be ensured. Thus, analysing these requirements and possible life cycle solutions we have selected evolutionary prototyping. This life cycle model is considered to be useful in situation where requirements are changing rapidly and users feedback must be frequently evaluated [11]. In the described project, the process of application developed was supported by groups of students, testing RIA after each step of

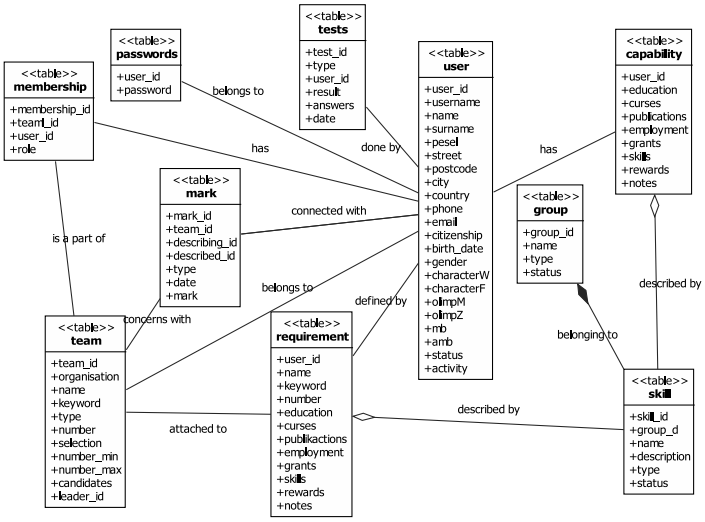

Fig. 4. The class diagram for TeamCreator (databases only)

functionality extension. All problems were reported to instructor and later to the developing team, influencing the next development steps.

The access to TeamCreator had to be possible from any web browser equipped with Flash plug–in. Therefore, practically usage of TeamCreator was not dependent on a hardware platform, operating system or web browser. We have tested it on Firefox, Internet Explorer, Opera and Safari browser, as well as on Windows and Linux OS. The application hasn't shown any platform specific problems. The appearance of presented DSS in Firefox 3 was shown on Fig. 6 (this screen capture is in the original form/language and was shown for GUI presentation purpose only). The majority of testing users highly appreciated that the used components in GUI were equivalent to the desktop ones. It the result, web–based system was not only easy to operate, but reminded a traditional desktop application.

During TeamCreator development we have observed some OpenLaszlo weaknesses e.g. as a still emerging technology LZX language contained significant errors (which were eliminated with frequent releases of OL). Moreover, we have observed important decrease in application performance with increase of its complexity. This problem was reduced by OL 4.2 release, that offered SWF9 support significantly increasing OL speed.

Despite these problems, the functionality, visual appeal and LZX language, supporting prototyping, kept our positive opinion about OpenLaszlo technology. It become a popular Open Source RIA technology and receives strong support of large developers community.

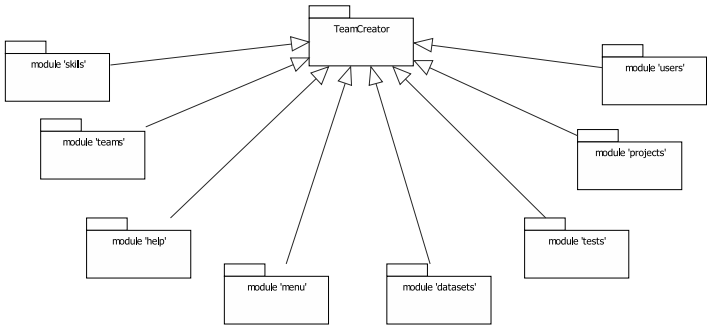

Fig. 5. The components diagram for TeamCreator

4 Sources of Risk for RIA development

Like all software projects also RIA development incorporates common risk sources, but there exists a group of sources particular to RIA. Due to it, some researchers focus on problems and risks connected with RIAs e.g. [12] divided the design of RIA onto four stages: data modelling, business logic modelling, presentation modelling, and communication modelling and introduced particular issues for each stage. As it is a wide topic, we would like to signalize only a few observed factors, that in our opinion are very important and may cause projects to be more exposed on risk.

The first of them is RIA oversizing i.e. web–services are build in size exceeding real requirements. Some desktop applications are said to be bloatware, but their over–sizing do not influence speed so much as we can observe for RIAs. The larger RIA is, the more data it must transfer and the more computer resources it uses. As RIA applications aren't compiled to native code and don't use resources so optimally, they are more heavy (in terms of speed not bytes) piece of software than comparable desktop application. Hence, if development does not take that into account, this mistake can lead to creation of slow application with a long start–up time and that will be opposite to users expectations. Of course, this threat is also important to desktop applications, but its significance is multiplied for RIAs.

Another important factor is preferably not to require any additional software available on users machine. Even if RIA uses technologically advanced and brilliant solution, it may not be successful when the most of its potential users will have to download and install additional, uncommon plug–ins. Therefore some vendors base their technologies on popular plug–ins e.g. Flex was based on Flash; OpenLaszlo was based on Flash or JavaScript; JavaFX was based on Sun JRE. Probably the future RIA technologies will not be focused on delivery of new plug–ins, but rather on shifting RIA languages to the fourth generation of programming languages.

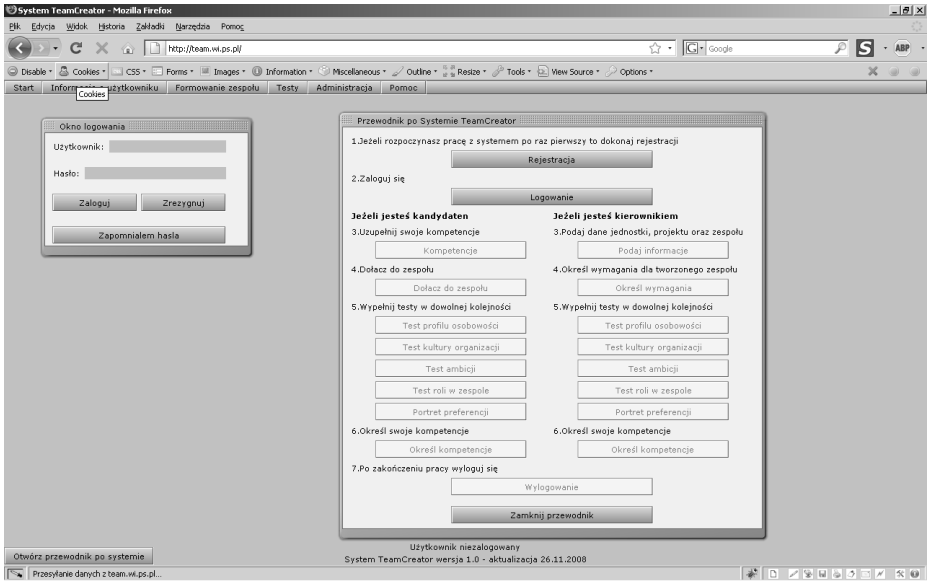

Fig. 6. TeamCreator start–up screen

It must be also remembered that skinning capabilities for RIA technologies should not be overused, as the visual attractiveness transforms into a eye–candy e.g. too extensive use of animation is a common mistake in Flash based systems. These technologies were invented to support creation of customised web–based applications. The visual design of such application should be consistent in all their parts. While the visual appeal helps in causing a positive impression, but the most important features attracting users are the content and functionality.

The last important factor, that in our opinion must be remembered during RIA development, is to supply project with more resources than comparable desktop projects. It's a result of RIA complexity, where often many different technologies are being used, required skills and environment are less homogeneous than for non–RIA projects. It terms of software engineering it can be said e.g. that one hypothetic function–point requires more work time for RIA than for desktop application.

5 Conclusions

The development of RIAs should be performed in a manner ensuring the major RIA quality factors. Analysing the quality factors we have focused on software usability, that in our opinion might positively distinguish these web–based applications. Fulfilling the usability factors the developed web–based software will be ergonomic, user–friendly and portable.

The distance between functionality of traditional desktop applications and RIA is decreasing. However, the advanced web–applications are exposed on new

sources of risk. In this article we have presented risk source that in our opinion are the most important ones. Finally, the scale, usability and ergonomics of Decision Support System described herein prove that RIA offers possibility to build advanced and user friendly web applications. As it was presented during system's introduction its design was focused on application quality and evolutionary prototyping, selected as a life cycle model, gave an ability to introduce user's remarks to the system.

Acknowledgements This work was financed as a part of Polish Ministry of Science and Higher Education research grant No N115 020 31/0713.

References

1. Newman, A., Steinberg, A., Thomas, J.:Enterprise 2.0 Implementation. McGraw–Hill Professional, New York (2008)
2. Casarez, V., Cripe, B., Sini, J., Weckerle, P.: Reshaping Your Business with Web 2.0: Using the New Collaborative Technologies to Lead Business Transformation. McGraw–Hill Professional, New York (2008)
3. Adobe – Flex 3, `http://www.adobe.com/products/flex`
4. OpenLaszlo — the premier platform for rich internet applications, `http://www.openlaszlo.org`
5. JavaFX — Rich Internet Applications Development — RIAs Java FX, `http://javafx.com`
6. Animation software, multimedia software — Adobe Flash CS4 Professional, `http://www.adobe.com/products/flash`
7. The Official Microsoft Silverlight Site, `http://silverlight.net`
8. java.com: Java + You, `http://java.com`
9. Klein, N., Carlson, M., MacEwen, G.: Laszlo in Action. Manning Publications Company, Greenwich (2008)
10. Olejniczak, W.:Team–Culture–Project. Wydawnictwo Zachodniopomorskiej Szkoy Biznesu, Szczecin (2009)
11. McConnell, S.: Rapid Development: Taming Wild Software Schedules. Microsoft Press, Redmond(1996)
12. Preciado, J., Linaje, M., Comai, S., Sanchez–Figueroa, F.: Designing Rich Internet Applications with Web Engineering Methodologies. In: 9th IEEE International Workshop on Web Site Evolution, pp. 23-30, Paris (2007)