XRX
The Implementation Process under XRX Architecture

Cristina Nemeş, Marius Podean and Lucia Rusu
Faculty of Economics and Business Administration, Babeş-Bolyai University, Cluj-Napoca, Romania

Keywords: Data model, XML, XPath, XQuery, XRX, XSLT.

Abstract: The XRX (XForms - REST - XQuery) architecture is a three tier architecture which uses at each tier data in XML format. This offers a great advantage because data is not being transformed in different other formats in order to communicate with other layers. Using this architecture the application becomes more agile, flexible and simple because there is no need of translations like in the classical architecture. This paper describes the implementation process of an application developed under the XRX architecture using W3C standards (XHTML, XML, XPath, XQuery, XSLT, XForms), REST interfaces and a native XML database.

1 INTRODUCTION

Nowadays every company works with databases and has to constantly improve the way they manage their data in order to facilitate CRUDS operations (Create, Read, Update, Delete, Search) on the database. Many software developers have implemented in relational databases the possibility to save data in XML (eXtensible Markup Language) format.

Databases represent a very important aspect in web applications because the exchange of data and documents inside a company and between companies is essential. To remain competitive on the market companies have to improve their way of exchanging data. Organizations have to exchange data and complex documents in the same format in order to be understood by other partners. Collaboration is an important process implemented by many companies in order to achieve common goals. An important issue is to ensure the exchange of complex documents between companies in a way that is software independent in order to facilitate the process of collaboration between organizations.

This paper presents a model that integrates these issues and brings some solutions. The paper presents a way of developing web applications under a new architecture called XRX. This architecture works with data in XML format at each tier unlike the classical architecture where data is stored in different formats. The aim of this paper is to make a comparison between the XRX architecture and the classical architecture and to emphasize the advantages of the XRX architecture. In order to achieve this goal we have implemented a web application according to XRX architecture.

A new idea for developing and implementing Web applications is based on W3C standards and REST interfaces. The W3C standards used are data format (XML, HTML, XHTML), programming languages (XQuery, XPath, XSLT), and schema (XML Schema) (Kaufmann and Kossmann, 2009). More and more web applications are developed using RESTful Web services because the complexity of developing applications that involve Web services access, data processing, and human interaction are reduced (Onose et al., 2009, Selonen et al., 2010).

XML is a language used especially on the WEB because it facilitates the exchange of data between applications which run on different platforms and are developed with different technologies. To provide the same data format we use a XML vocabulary called UBL (Universal Business Language) (Bosak and McGrath, 2006) which is a standard for economic documents like orders, products catalogs and invoices.

In section 2 we present the collaboration concept, and in section 3 we provide a brief overview on the XML stack of languages used for implementing the application and XRX architecture. The application is described in section 4. Section 5 presents related works. In section 6 are presented conclusions and future work.

Nemeş C., Podean M. and Rusu L..
XRX - The Implementation Process under XRX Architecture.
DOI: 10.5220/0003931101030109
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 103-109
ISBN: 978-989-8565-08-2

2 COLLABORATION

The process of collaboration has emerged due to the need of communication between companies.

Collaboration is a process in which entities share information, resources and responsibilities to achieve a plan, implement and evaluate a program of activities to achieve the same goal (Camarinha-Matos and Afsarmanesh, 2008). In order to collaborate, organizations have to exchange data and documents in the same format in order to be understood by other partners. To work, organizations need a common language that will enable the implementation process with minimal effort.

In terms of computing, collaboration is the integration of different technologies into a single platform in order to improve the performance of data and complex documents exchange. The model for the common language in our case is UBL which is based on XML. UBL is used in order to create complex documents that have the same format so that organizations that use the same standard can easily exchange data.

UBL is the language used in order to implement the process of collaboration in our application. In the following section we describe the XRX architecture and the W3C standards used in order to develop the application.

3 XML FAMILY AND XRX ARCHITECTURE

This section presents the key elements of the XML stack of languages and the XRX architecture used in the implementation process.

XRX is a "no-translation" (McCreary, 2010) web application architecture based on three technologies (McCreary, 2008):

- XForms on the client;
- REST interfaces;
- XQuery on the server.

According to (McCreary, 2007) this architecture gives web developers a tenfold increase in productivity.

XForms is an XML application that separates content from presentation. After (Boyer, 2007) XForms is not a document type to be used independently so it has to be used with XHTML or SVG, and (Turner and Windauer, 2010) states that XForms will become the next generation of forms for World Wide Web. XForms is based on the MVC (Model-View-Controller) architecture and is compatible with other W3C standards like CSS, XPath, XQuery, XML Schema. More and more companies (Semanta, Volkswagen Financial Services, Australian Bureau of Statistics, Teleflex, Manugistics) want to enjoy the XForms benefits so they are collaborating with Orbeon in order to obtain very efficient forms (Orbeon, 2012).

REST is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP, WSDL). Many websites like Twitter, Yahoo, Flicker, Vimeo, PhotoBucket, LinkedIn, Amazon S3 use the REST protocol due to its great characteristics. (Fielding, 2000) defines REST as a set of architectural constraints that emphasizes scalability of component interactions, generality of interfaces, independent deployment of components and enforces security.

XQuery is a query language which allows to select information according to one or more criteria, joining data from multiple documents, mathematical operations on numeric data type (Walmsle, 2007). Instead of using XQuery we can use XSLT. XSLT (Clark, 1999) is a declarative language used to transform XML documents into text, HTML or XML. An XSLT stylesheet describes the rules that transform a source tree into a result tree. XSLT uses XPath expressions to select the source nodes from the trees. XSLT is used in general for the presentation of data, while XQuery is much more for making queries and working with data. XSLT can be used instead of XQuery but the code is not so compact like with XQuery.

XML is a technology used to represent data and documents. By using XML technologies, a high portability of documents can be obtained (Bray et al., 2008). XML allows to define vocabularies that describe data according to an activity domain in order to facilitate the exchange of documents between business partners from the same area. UBL is a synthesis of existing XML B2B languages (RosettaNet, cXML, xCBL) that is interoperable with EDI systems. It can be applied in different domains like administration, electronic trade.

All these languages are used in the implementation process described in the next section.

4 APPLICATION

XRX architecture is a new three tier architecture which offers the possibility to develop complex applications. This architecture can very easily be combined with the MVC (Model-View-Controller) pattern. In our case the Model is stored in the eXist

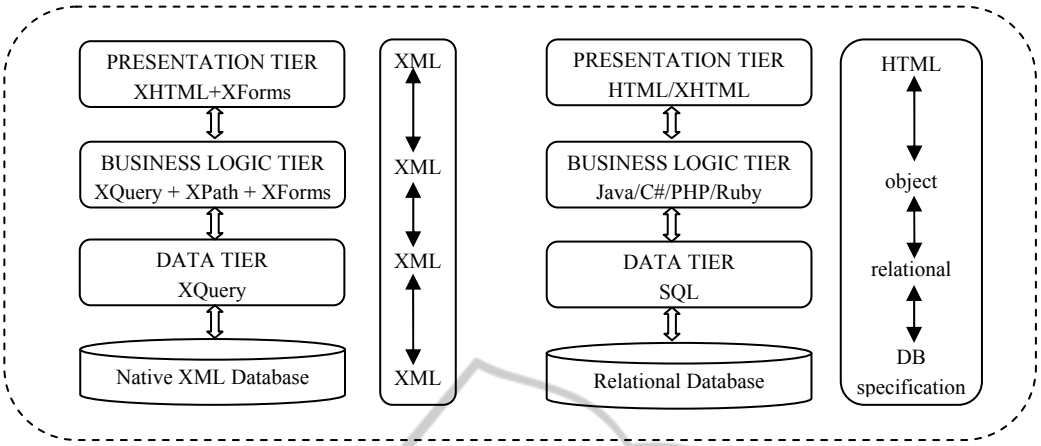

Figure 1: XRX architecture and traditional architecture.

database, the View is generated automatically from XForms elements and the Controller part can be implemented in XQuery/XSLT/XProc or as a combination between these technologies.

4.1 Application Architecture

The application developed under the XRX architecture uses at each tier W3C standards, that work with data in XML format (Figure 1).

In this architecture there are no translations between the languages used at each tier because all work with data in XML format. On the other hand in the classical three tier architecture which uses HTML in the browser, Java/C#/PHP at the business logic tier and SQL databases there are needed four types of translations (McCreary, 2010):

- HTML to objects;
- objects to SQL;
- SQL to objects;
- objects to HTML.

To develop an application under the classical architecture is required hard work to construct, test and maintain the application, and (Davis and Maguire, 2011) calls the complex set of transformation "non-value-added transformations". The XRX architecture is called a “Zero Translation” (McCreary, 2010) architecture because there is no need for transformations between the tiers.

On the presentation tier, which can be seen as the View in the MVC pattern, we use XHTML and XForms in order to display the graphical interface to the user in the browser instead of the classical HTML forms. XForms is very effective in processing sets of complex semi-structured data. In the classical applications JavaScript is used to make validations on the client, to communicate with the server and also to make some calculations. JavaScript is an object oriented language which has a syntax influenced by that of C (Goodman, 2001). For very complex and dynamic user interfaces XForms can be extended with JavaScript. We use XForms instead of JavaScript in order to process complex datasets, to make validations on the client, calculations and constraints. A very important advantage of XForms is that it processes very well documents in different formats. Working with JavaScript for processing and manipulating documents in XML format requires many lines of code for inserting, deleting and making some changes in the XML trees. In Figure 2 is presented an extract from a JavaScript function which creates an XML tree using DOM methods.

At the business logic tier we use XQuery and XPath to manage data in XML format instead of PHP, Ruby or object oriented languages like Java and C#. In (Costello, 2008) a problem is solved in two ways: using XSLT and using Java. The result of his experiment was that the problem was solved with XSLT in 10 lines of code and over 100 lines in Java. In Figure 3 and 4 we present the code for transforming the order in order-response using an XSLT stylesheet. These figures show that is much more easier and simple to implement this transformation using XQuery instead of PHP.

In a classical architecture at the data tier is used SQL to manage data stored in relational databases. We use XQuery to manage data stored in XML format in the eXist database, which is a native XML database. Relational databases contain tables where data is stored as records. A native XML database contains collections of XML documents.

```
function createXMLDocument(xhr)
{
        response=xhr.responseXML
        root=response.documentElement
        buyer=document.createElement("buyer")
        buyer.setAttribute("id","Th01")
        name=document.createElement("name")
        contentname=document.createTextNode("Thomas")
        name.appendChild(contentname)
        buyer.appendChild(name)
        address=document.createElement("address")
        contentaddress=document.createTextNode("London Street")
        address.appendChild(contentaddress)
        buyer.appendChild(address)
}
```

Figure 2: Creating a XML document using JavaScript.

```
xquery version "1.0";
declare namespace transform="http://exist-db.org/xquery/transform";
let $order:=doc("/db/betterform/ubl2/order.xml")
let $orderresponse:=doc("/db/betterform/ubl2/orderresponse.xsl")
let $transform:=transform:transform($order,$orderresponse,())
return $transform
```

Figure 3: Transforming an XML document with XQuery.

```
<?php
     header('Content-type:application/xml');
     $order=new DOMDocument();
     $order->load('order.xml');
     $orderresponse=new DOMDocument();
     $orderresponse->load('orderresponse.xsl');
     $processor=new XSLTProcessor();
     $processor->importStylesheet($orderresponse);
     $transform=$processor->transformToXML($order);
     print $transform;
?>
```

Figure 4: Transforming an XML document with PHP.

4.2 Application Description

In this section we describe the application explaining the content and how the user has to interact with it.

Our application implements the exchange of economic documents in electronic format between different companies. The application is composed of a wizard which represents the order and two documents: order response and the invoice.

On the first page of the wizard presented in Figure 5 are displayed the products from which the client may choose. The client has to activate the "Add" button corresponding to the desired product. When the button is pushed below the list of products appear the selected one with the implicit quantity 1. The quantity can be changed by typing another, and in order to remove a selected product the customer has to type “0” in the quantity field.

On the second page are displayed in a table details about the customer like full name, address, work-address, e-mail.

On page three presented in Figure 6 the customer selects the delivery address. The customer has to select a radio button which corresponds to the desired delivery address. Also the customer can add

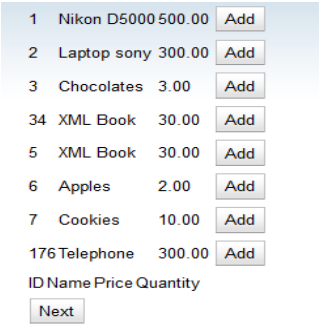

Figure 5: First page of the wizard.

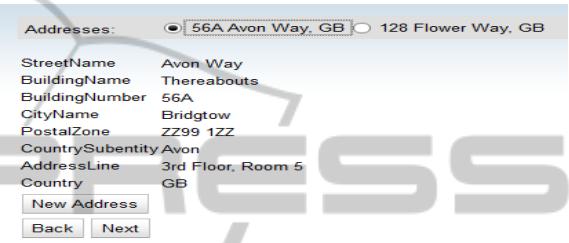

Figure 6: Selecting the delivery address.

a new address, and the page is dynamically updated (appears a new radio button for the new address).

The same process applies when choosing the method of payment except that the client cannot introduce a new payment mean.

To choose the delivery date the client has to choose the date using a date picker.

The last page of the wizard presents a table of selected products that the client wants to order. Are displayed the product ID, name, quantity, price and are calculated the value of the products (price * quantity) and the VAT (value of products * percent VAT). Finally, is calculated the total amount of the order as the sum of value of goods and VAT .

At the final step the buyer has to push the "Submit" button to send the order. If the order has been submitted successfully the seller receives an order response that specifies if the order is accepted or not. If the order has been accepted the buyer will receive an invoice for the purchased goods.

Each step in the wizard is mandatory. If a customer does not complete a step or the introduced data is invalid the customer receives an information message. For example Figure 7 presents a message validation informing the user that the selected delivery date is invalid because it must be greater than the start date which is the current date or the date of the order. If a client fails to complete one or more steps in the wizard when the "Submit" button is pressed to send the order, the order will not be sent (Figure 8).

In this section we presented the interaction of the user with the application by describing each step of the wizard. In the next section we present the implementation process.

4.3 The Implementation Process

In this section we present the implementation process for our web application developed under the XRX architecture. This application works with data in XML format in order to improve the exchange of information between partners according to the UBL standard.

The application is implemented under the XRX architecture and also according to the MVC design pattern. The model contains the data values, the view is represented by the screen presentation and the controller defines the way the user interface reacts to user input (Gamma, 2009). MVC can use other design patterns like Decorator to add scrolling to a view or Factory Method in order to specify the default controller class for a view (Gamma, 2009).

The model is composed of the XML documents stored in the eXist database. The XML documents in the database are validated according to the UBL standard.

The view represents the user interface which is composed of XForms elements and XHTML. The view is the wizard and the two documents: order response and invoice. The wizard is generated dynamically from a query. Also the documents are generated dynamically from an XQuery code using different eXist specific functions in order to save the data completed by the user in the database, and make the XSLT transformations from order in order response, and then from order response in invoice.

The wizard is created using a switch-case structure where each case represents a page/step of the ordering wizard. The client has the possibility to navigate back and forward to make some changes using "Back" and "Next" buttons which appear on each page of the wizard. To implement a button we use the trigger element. A trigger is an abstraction of a web button that allows forms to be more portable.
To choose the delivery date we created an input type element that allows the customer to choose a date. Because through a bind element we specified the type of this element as type = "xs:date" at a simple click in this box appears a calendar that allows the user to choose the date (date picker).

The controller part is composed of the XQuery documents that generate the wizard and use the eXist specific functions to save the customers data in the database, and to make the XSLT

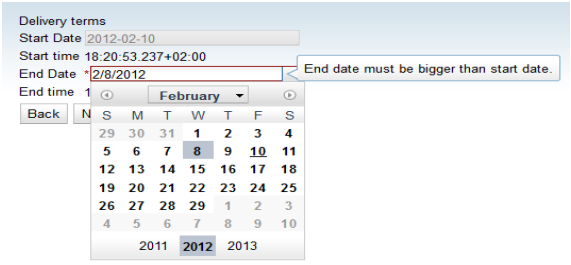

Figure 7: Delivery date validation.

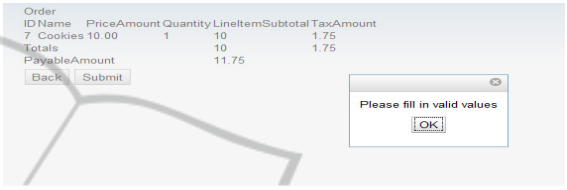

Figure 8: Error validation in the wizard.

transformations.

We have created a module where we have implemented the necessary functions that generate the wizard. The function businessrules presented in Figure 10 gets all the child elements from the "bind.xml" document. In this document are implemented the bind XForms elements. The function products selects all the "cac:OrderLine" elements from the "products.xml" document. This function selects all the products stored in the xml document.

In the classical architecture instead of XQuery is used PHP, Java, or other languages. When working with PHP in general is used a relational database like MySQL. Making a comparison between the code for extracting the data using XQuery and using PHP we can say that using XQuery is more efficient and simple than using PHP.

According to (McCreary, 2008) the number of code lines when using XQuery is much smaller than the number of lines of code when using SQL and Java.

(Candillon, 2011) presents a comparison that shows the amount of code lines needed to develop the same application using different technologies like Java, PHP and XQuery. The charts present that using XQuery the number of lines of code can be

```
<xf:trigger>
    <xf:label>Back</xf:label>
    <xf:toggle case="pag4" ev:event="DOMActivate"/>
</xf:trigger>
<xf:trigger>
    <xf:label>Next</xf:label>
    <xf:toggle case="pag6" ev:event="DOMActivate"/>
</xf:trigger>
```

Figure 9: Back and Next buttons.

```
declare function modul:businessrules() as node()*
{
        let $business_rules:=doc("bind.xml")/*/child::*
        return $business_rules
};
```

Figure 10: Function businessrules.

reduced up to 80% if we refer to Java or with 62% if we refer to PHP (Figure 11).

The PubZone application was implemented in Java and in XQuery. The XQuery implementation ended up to be more compact and elegant than the version in Java (Kaufmann and Kossmann, 2009, Candillon, 2011) (Figure 12).

In order to develop complex web applications using the XRX architecture is needed only one person who can do the work of ten people (McCreary, 2010). When developing an application under a different architecture many persons are needed in order to implement the application using different technologies. Thus the costs of developing an application using the XRX architecture are smaller than in the case of other type of architecture (McCreary, 2010).

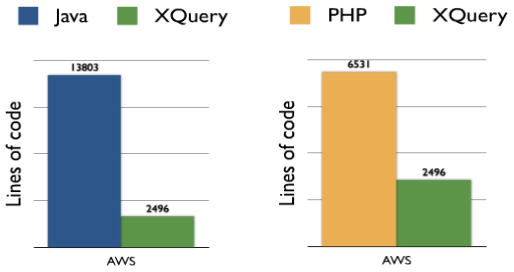

Figure 11: Java vs. XQuery and PHP vs. XQuery (Candillon, 2011).

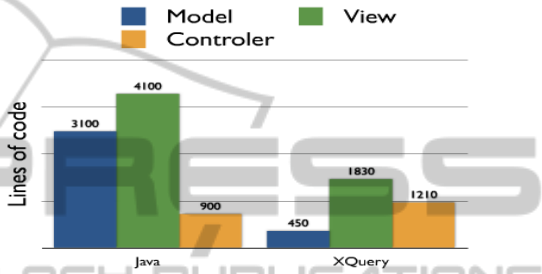

Figure 12: Java vs. XQuery lines of code - PubZone (Candillon, 2011).

5 RELATED WORK

Previous work in the field of REST like (Fielding, 2000) has a great impact on the applications that use this architectural style. Also some important works related to XML, REST and XQuery are (Kaufmann and Kossmann, 2009) and (Davis and Maguire, 2011) which have developed applications using XML technologies.

Regarding the XRX architecture the most relevant and detailed explanations about this architecture are given by (McCreary, 2007), (McCreary, 2008) and (McCreary, 2010).

(Hunter, 2012) presentes the open source "Corona" project which uses the MarkLogic Server in order to manage and query XML and JSON using REST. (Retter, 2012) presents a research regarding the current approaches for invoking XQuery in a RESTful manner over HTTP and the standardised XQuery 3.0 annotations for REST. In (Couthures, 2012) and (Lenz, 2012) are presented different approaches that are using XQuery and XSLT in order to develop new applications.

6 CONCLUSIONS

The application presented is developed using XML, XQuery, XSLT, XForms and UBL standard used to represent economic documents in electronic format. The application is built on three levels according to the XRX architecture. This architecture replaces the traditional three-tier architecture that uses HTML in the browser, on the logical level languages such as PHP, Ruby, C #, Java and SQL for relational databases.

The comparison presented between the XRX architecture and the classical architecture shows that applications developed according to the XRX architecture are more efficient and elegant. Moreover the XRX architecture increases productivity and reduces the implementation costs. In our research one of the main goals was to develop an application that can ensure a good exchange of data between companies in order to facilitate the process of collaboration.

In the future we plan to develop an application following the same principles but using XProc (XML Pipeline Language). XProc is a language used to create pipelines which takes zero or more XML documents as input and produces zero or more XML documents as output. An important advantage of XProc is that allows other technologies from the XML stack of languages to interact (XML Schema, XInclude, XQuery, XSLT).

REFERENCES

Anders, B., Scott, B., Don, C., Mary, F., Michael, K., Jonathan, R., Jerome, S., 2010. XML Path Language (XPath) 2.0 (Second Edition). W3C Recommendation 14 December 2010 http://www.w3.org/TR/xpath20/.

Binemann-Zdanowicz, A., Schewe, K.D., Thalheim, B., 2005. Development of Collaboration Frameworks for Distributed Web Information Systems. In *Proceedings of iiWAS'2005*. pp.551-562.

Bosak, J., McGrath, T., 2006. Universal Business Language v2.0. http://docs.oasis-open.org/ubl/os-UBL-2.0/UBL-2.0.pdf.

Boyer, J. M., 2007. XForms 1.0 (Third Edition). W3C Recommendation 29 October 2007 http://www.w3.org/TR/2007/REC-xforms-20071029/.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F. 2008. Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation 26 November 2008 http://www.w3.org/TR/xml/.

Camarinha-Matos, L. M., Afsarmanesh, H., 2008. Collaborative Networks: Reference Modeling. http://www.springerlink.com/content/978-0-387-79425-9#section=209705&page=7&locus=31.

Candillon, W., 2011. Not your Grandma's XQuery. http://www.slideshare.net/wcandillon/not-your-grandmas-xquery.

Clark, J., DeRose, S., 1999. XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999 http://www.w3.org/TR/xpath/.

Clark, J., 1999. XSL Transformations (XSLT) Version 1.0. W3C Recommendation 16 November 1999 http://www.w3.org/TR/xslt.

Costello, R., 2008. XML versus Data Binding. http://www.xfront.com/xml-versus-data-binding/index.html.

Couthures A., 2012. Compiling XQuery code into Javascript instructions using XSLT Exploiting XQuery grammar. In *Proceedings XML Prague 2012*. pp. 125-139.

Davis, C., Maguire, T., 2011. XML Technologies for RESTful Services Development. *Proceedings of the Second International Workshop on RESTful Design*.

Denise, L., 1999. Collaboration vs. C-Three (Cooperation, Coordination, and Communication). In *Innovating*, Vol.7, Nr.3. http://www.ride.ri.gov/adulteducation/Documents/Tri%20part%201/Collaboration%20vs.%20the%203c%27s.pdf.

Fielding, R., 2000. Architectural Styles and the DEsign of Network-based Software Architectures. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Fuks, H., Raposo, A., Gerosa, M. A., Pimental, M., Lucena, C. J. P., 2008. Encyclopedia of E-collaboration. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, ch. The 3C Collaboration Model, pp. 637-644.

Gamma E., Helm R., Johnson R., Vlissides J., 2009. Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. pp. 4-6.

Goodman, D., 2001. JavaScript Bible 4th Edition. Hungry Minds, New York.

Hunter J., 2012. Corona: Managing and Querying XML and JSON via REST. In *Proceedings XML Prague 2012*. pp. 73-80.

Kaufmann, M., Kossmann, D., 2009. Developing an Enterprise Web Application in XQuery. In *International Conference on Web Engineering*.

Lenz E., 2012. Implementing an XQuery/XSLT hybrid Parsing and compiling Carrot. In *Proceedings XML Prague 2012*. pp. 141-170.

McCreary, D., 2007. Introducing the XRX Architecture: XForms/REST/XQuery.http://datadictionary.blogspot.com/2007/12/introducing-xrx-architecture.html.

McCreary, D., 2008. XRX: Simple, Elegant, Disruptive. http://www.oreillynet.com/xml/blog/2008/05/xrx_a_simple_elegant_disruptiv_1.html.

McCreary, D., 2008. XRX: XForms, REST and XQuery Simple, Elegant, Disruptive.http://www.danmccreary.com/training/xrx/index.html.

McCreary, D., 2010. Using Native XML Systems to Manage Metadata. http://www.tdan.com/view-articles/14517.

McCreary, D., 2010. The National Information Exchange Model and Semantic-Driven Development. http://semanticweb.com/files/SU/NIEM-Slides-v2.pdf.

Onose, N., Khalaf, R., Rose, K., Siméon, J., 2009. A Restful Workflow Implementation on Top of Distributed XQuery.

Orbeon 2012.http://wiki.orbeon.com/forms/welcome/sites-projects-companies-using-orbeon-forms.

Retter A., 2012. RESTful XQuery Standardised XQuery 3.0 Annotations for REST. In *Proceedings XML Prague 2012*. pp. 91-123.

Selonen, P., Belimpasakis, P., You, Y., 2010. Developing a ReSTful Mixed Reality Web Service Platform. In Proceedings of the *First International Workshop on RESTful Design*. pages 56-63.

Vintilă, B. 2010. Collaborative Applications in the Knowledge Based Society. In *Journal of Applied Collaborative Systems* Vol. 2, No. 1.

Walmsle, P., 2007. *XQuery*. O'Reilly Media, Inc.

Turner, J., Windauer, L. 2010. betterFORM User Guide. http://www.betterform.de/doc/betterFormUserGuide.pdf.