Paperful to Paperless Office Forms Integration Framework

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment of the Requirement for the Degree

Master of Science

By

Madhuri Potu

May 2004

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Christopher Pollett

Dr. Agustin Araya

Dr. Stamp Mark

APPROVED FOR THE UNIVERSITY

Abstract

Forms are one of the most important transactions on the web. They enable the users to conduct transactions in a simple and efficient manner. It is the most efficient way of gathering and processing the information from the user. There are many examples of uses of Forms on the web: surveys, online trading, purchasing. Most of the forms are written in HTML which is or was our solution to web applications. Advancements in technology have replaced HTML by a more reliable and independent technology called XForms. They are considered to be the future of web applications.

This project develops electronic forms in a framework; it lets the user design his/her own forms, edit them, merge any two forms and can be integrated into applications that store the data. In addition it can even maintain a log of revision data, i.e., a history of the revisions made to the forms – the date it was created or modified and a description of the modification. The framework has been developed using Xforms, XSLT, Javascript, JSPs, and the forms are stored in XML Format in an Oracle database. The need of usability testing has been addressed by following Usability Patterns like System feedback, Data Validation, History Logging and User Profiles.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

Transactions over the Internet such as online banking, filling out surveys, and email have become a part of our daily routine. Using Forms is one of the most efficient ways of collecting and processing information over the Internet. HTML is the general format for creating forms. It has predefined tags that let you describe data only once. On the other hand, XML (Extensible markup Language), an advancement in technology, provides more flexible and structured information, lets users customize their applications thereby helping to improve the functionality on the web. XML is not a replacement to HTML it is more of an extension to HTML. XML is best used in describing the data with user defined tags while HTML is more suited for formatting and displaying the XML data.

The next stage in the development of HTML forms is XForms. The biggest advantage of XForms is that the data and logic are separated from the presentation. They use XML to describe the data and HTML to display the data. This advantage makes them work on any platform; since the data model can be separated, it can be made to work with any interface, such as handheld devices, phones etc. XForms help us to overcome the limitations of HTML forms by using XML. Various advantages of XForms over HTML forms are described in the following chapter.

The framework developed in this project lets users create, edit and customize forms. The forms contain a set of controls like textfields, drop down menus, and checkboxes, that can be used to enter information. Generally, for web transactions once the information is submitted it is then transmitted over the Internet to the appropriate place. It is then processed and may also be stored in a database so that redundancy does not occur. We can establish relationships between data using either Relational Databases or XML documents. In this project we use an Oracle database to store the information about the elements in the XML Document; the forms are stored as XMLType Tables [Oracle XML DB, Chapter 2]. This way of storing the data that lets you insert XML type documents into the database without removing the relationships between the data is called Structured Method. Both relational databases and XML documents can be combined in such a method. The *Document Object Model [DOM] [11]* is used for accessing the XML documents that is to add new elements, to delete or modify elements. DOM can access the document once it's been parsed by a parser [ex: SAX Parser]. DOM is platform independent and can be used with any programming language.

A Version Control System has been implemented to maintain the revision history of the Forms as well as that of a user. The system contains all the information regarding the date and time of the creation of the form, what modifications were made to the form, and what revisions a form has undergone. When two forms are merged to create a new form, the information about the ancestors is stored in addition to the changes made to them. Usability patterns were implemented to improve the quality of the application. History logging (Version Control System maintains all the history), data validation are some of the patterns implemented in the project. A log of the information regarding the last page visited by the user during his/her last login is made, therefore helping the user remember what he/she was working on.

This report is organized as follows. Chapter 2 describes the background and related work on XML, XForms, JSPs, JavaScript and Oracle XML DB. Chapter 3 gives a description of the design and implementation of the office Forms Integration Framework. In the design we look at Pattern programming, Application Security and Usability patterns that were implemented to increase the usability of the user. Chapter 4 describes the usability testing done and their results. Chapter 5 discusses the conclusion and future enhancements.

Chapter 2

Background and Related Work

This section discusses the technologies used in this project i.e., XML, XSLT, XForms, Jsps, Javascript, Oracle, DOM and SAX parser.

2.1 XML

XML (Extensible Markup Language) is similar to HTML. However, XML does not have predefined tags like HTML; we can define and customize our own tags. XML is used to describe the data and is independent of presentation. This is the most important difference between XML and HTML. *XML--- describes data, HTML – displays that data*. XML describes data in such a manner that anyone can work on it and a person later on referring to it also can understand it easily. The example below describes the growing importance of XML: [12] [www.congressonlineproject.org/glossary.htm]

“The House of Representative has recently issued a list of XML tags to be used in web forms on Member/Non Member websites that send email to congressional offices. The purpose of the forms is to enable Correspondence Management System (CMS) to easily identify and process type of information – such as name, city, zipcode, etc. which will help make the software efficient and more effective.” From this example we can strongly support the statement that XML will be integral to the future of the web as

HTML has been to the foundation of the web. XML will become the most common tool for all data manipulation and data transmission.

2.2 XSLT

XSLT (eXtensible Stylesheet Language Transformations). The main purpose of XSLT is to transform XML documents to another form i.e., HTML or another XML document. In this project an XSLT stylesheet was used to transform the XForm documents and also to apply the presentation to the document. For example whenever an "input type" element occurred in the form, the stylesheet applied the following style.

h1 {color:"blue";font-family: \"Arial\"font-size:"12"px; }; -- This is extracted from the style information stored in the XForm using the following tag

<style type=\"text/css\" MEDIA=\"screen, print\"><xsl:value-of select= \"html/layout/fontsize\" />

As you can see in the above example the font size and color are not predefined. The user gives them. So the user can decide color and size of the font. It also decides the order in which the elements should be displayed. XSLT uses XPath in the transformation process. For example, consider **"<xsl:value-of select= form1/layout/inputbox"/>".** In this expression XPath finds matches in the source code to the path described above so that XSLT can transform that matching part of the document into the new document.

2.3 XForms

XForms are the next generation of web Forms. XForms separate the data from the presentation. This makes the XForms platform independent. E.g: can be used on desktop, PDAs, paper etc. XForms try to overcome the limitations of HTML Forms. For Example: The data in XForms is described using XML -- this integration with XML was not possible with HTML -- even for validation. XForms also send the data using XML i.e., data transmission over the internet. All the forms created in this project are written in XForms using XML to describe the data.

The following is an illustration of XForms model:

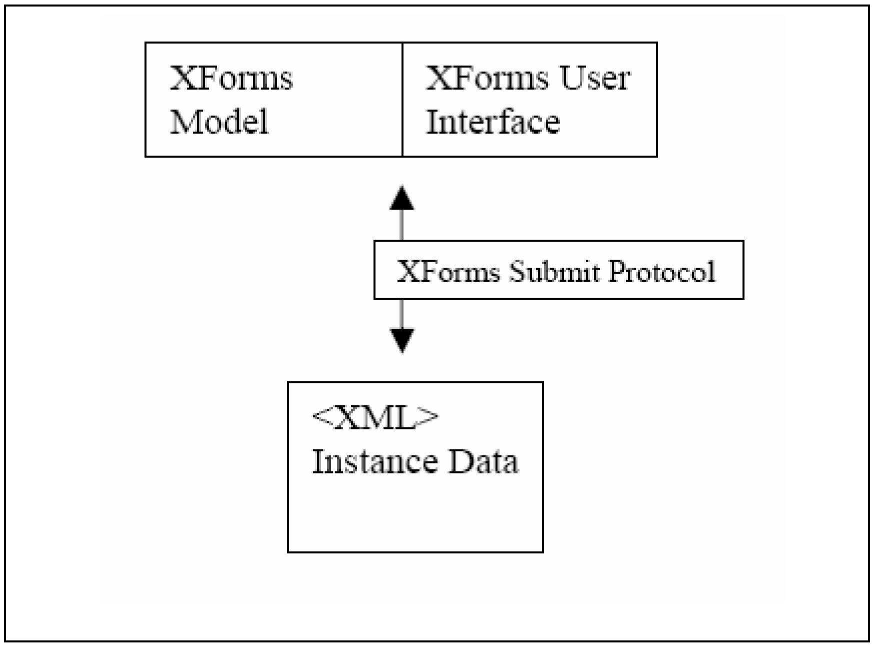


Figure 1: XForms Model

Source: http://www.w3.org/MarkUp/Forms/

XForms consists of three parts: User Interface, XForms Model, and Instance Data. The XForms model can work with any type of User interface – XForms User Interface, XHTML. The XForms Model describes the data in the form. The presentation is separated from the data and described in a separate section. The XForms User Interface has a set of controls that can be used in any type of device or platform. There are binding attributes to bind these controls with the XForms Model, e.g: ‘reference’ attribute. The Instance Data is the data submitted to or collected by the forms; this data can be sent or received using the XForms Submit Protocol and can be directly submitted by the XML processor.

An example XForm is described below:

```
<head>
<xforms:model>
<xforms:instance>
        </name>
</xforms:instance>
</xforms:model>
</head>
<body>
        <xforms:input ref="name">
        <xforms:label>First Name:</xforms:label>
        </xforms:input>
</body>
```

Figure 2: XForms Example

The ‘xforms:model’ element is used to represent the information needed in a form (in the above example: name). All this information is contained within the head tag. The content part is separated from the presentation making it easier for the forms to get only the data that is needed. XForms contain a set of controls that are described in the body section, these controls are bound to the 'xforms:model' element via the 'ref' attribute.

In the above example we described a form control called "xforms:input", this control describes an input box and the 'xforms:label' associated with it is the label of the input box. Since we will only be representing text boxes, combo boxes and submit buttons in this project, we will deal only with those controls. The 'xforms:selectone', 'xforms:choices', 'xforms:item' are the controls used to represent combo boxes and the 'xform:submit' is the control used for the Submit button.

Here is an example showing the form control for combo boxes (choice for gender male or female):

```
<xform:selectOne xform="entryform" ref="gender">
<xform:caption>Sex</xform:caption>
      <xform:choices>
              <xform:item value="male">
              <xform:caption>Male</xform:caption>
              </xform:item>
              <xform:item value="female">
              <xform:caption>Female</xform:caption>
              </xform:item>
      </xform:choices>
   </xform:selectOne>
```

Figure 3: xforms control for comboboxes

The 'xforms:selectone' does not just represent combo boxes, it can either be check boxes, radio buttons or combo boxes, depending on the person choosing it. 'xforms:selectone' only describes that we can choose any one among the given choices (combo boxes or check boxes or radio buttons), giving the user the freedom to select any type of representation. An XForms Processor is used to validate the data and send it directly to the application. It is a built in processor enabled in browsers.

With the data separated from the presentation, XForms can be managed efficiently, making them device or platform independent. In the future all browsers will be enabled with the XForms plugin making the users not feel the difference (between XForms and HTML forms) except they will find more efficient web forms. This information is used to develop a program that can create XForms. Once the XForm is developed an XSLT stylesheet is used to transform the xform document. The XSLT stylesheet used in this project “formstylesheet.xsl”, can be used to transform any type of XForm document that contain textfields, radio buttons, checkboxes, etc.
The basic idea of generating XForm Document and then transforming it using XSLT can be shown in the diagram given below:

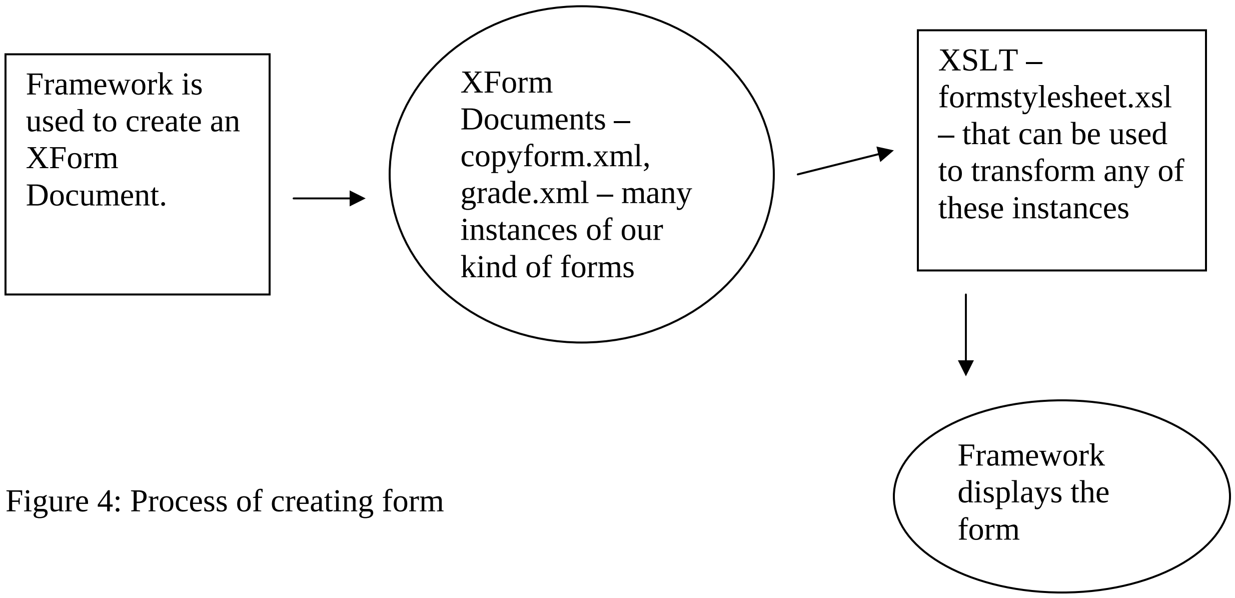

Framework is used to create an XForm Document.
XForm Documents – copyform.xml, grade.xml – many instances of our kind of forms
XSLT – formstylesheet.xsl – that can be used to transform any of these instances
Framework displays the form

Figure 4: Process of creating form

In the framework we create an XForm document with the tags and layout as shown in the diagram given below. The layout information and data is stored in the XForm document. Once the document is created “FormStyleSheet.xsl” is used so that it can be displayed in the framework. All the information regarding the layout such as the size, height and width of the “input tag” is stored in the “layout” tag. It basically describes the style of the form. “FormDocument” is the tag where the information about the xform like textfields, check boxes is stored. It contains model, instance and a control group to describe the form controls

```
<layout>
      <fontsize>
            h1 {color: blue ; font-family: "Arial", serif;
            font-style:italic, oblique; font-size:15px; }
      </fontsize>
</layout>
<formdocument>
      <model>
            <instance>
                  <title/>
                  -------
            </instance>
      </model>
      <controlgroup>
            <input ref="FirstName">
                  <caption>First Name</caption>
            </input>
            --------
      </controlgroup>
</formdocument>
```

Figure 5: Layout of the Xform

2.4 JSPs and JavaScript

JSP --- Java Server Pages is used for developing dynamic Web pages. JSP creates HTML and XML pages that combine fixed page templates with dynamic content. In this project we used JSP to generate web pages by combining XML and Java. JSP is a server-side programming language whereas JavaScript is a client-side programming language.

JavaScript is a scripting language that allows one to add dynamic content to HTML pages for enhancing Web pages. For example: to display forms and textfields. It is totally different from Java even though it shares some features. JavaScript has small functions enclosed in scripts so that the user can use it in the right way. In this project we used JavaScript functions to control the mouse movements and record the mouse positions to support the design and organization of forms. An example script is given below:

```
<script>
function mousemove(e)
{
        if (doDrag)
        {
                if (!e) {e = window.event}
                var oSource = window.event.srcElement
                if(oSource.name=="textfield")
                o=document.getElementById("Object1")
                ------
        }
}
</script>
```

Figure 6: Example JavaScript

2.5 Oracle XML Database

The Oracle XML Database provides standards and methods for navigating, storing and retrieving XML. Oracle helps us store, update, delete, query XML data and the same XML data can also use SQL (Structured Query Language) i.e. "*it encompasses both Structured Query Language and XML data model in a highly interoperable manner that provides native XML support*". [Oracle 9i] In brief this lets us use both relational databases and XML documents at the same time.

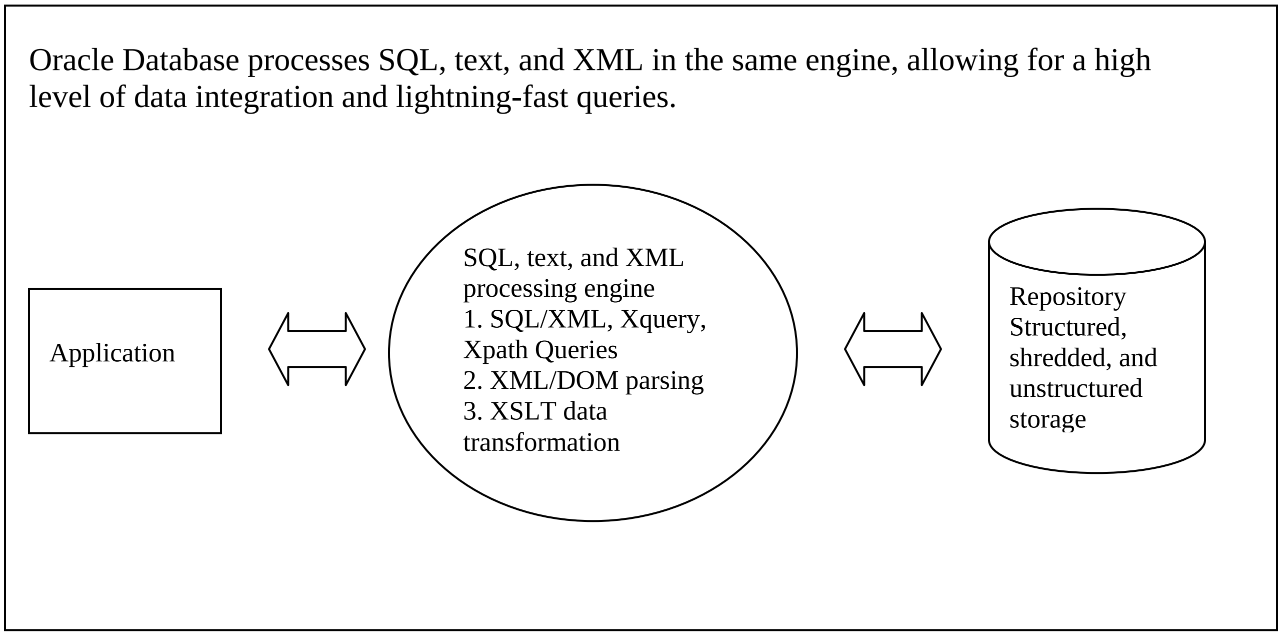

Oracle Database processes SQL, text, and XML in the same engine, allowing for a high level of data integration and lightning-fast queries.
Application
SQL, text, and XML processing engine
1. SQL/XML, Xquery, Xpath Queries
2. XML/DOM parsing
3. XSLT data transformation
Repository Structured, shredded, and unstructured storage

Figure 7: Oracle XML DB
Source: Oracle 9i

The following are the three methods used in storing XML data in relational databases:

Shredded, UnStructured, and Structured [5]

Shredded: The Shredded method does not store the relationships in the XML Data i.e., it does not store the entire XML document, it uses relational columns to store the data, removing the XML type i.e. it uses XML Type Column to store the XML Data. The format of the XML data is lost in this method. Consider the example where information needs to be transferred to different types of databases. It is shredded so that we can send the data wherever we want in an efficient manner.

Unstructured: This method stores the data using a *CLOB (Character Large Object)* to store the XML document. It retrieves the relationships of the XML data, i.e., the structure of the XML document is preserved. The only problem with this is the relational database queries cannot be applied on this type of data. A best example for this type of method would be a place where all the originals need to be in XML, i.e., it lets us create a relational record for the original and the original can also be stored with the record.

Structured: The last method lets the user preserve the structure of the XML document as well as apply relational databases on the data. In our project we use this method to store the XML data. We store the document using XML Type Table. The main advantage of preserving the relationships between XML data is being able to create an XML Document by manipulating the XML data with relational data.

Storing Data in XML Type Table:

Creating XMLType Table:

```
Create table xforms(form xmltype, formn varchar2(50), version number, id number, root1
varchar2(50), descendants number, root2 varchar2(50), sdate varchar2(30));
```

Figure 8: Create XML Type Table

Inserting Values in XMLType Table:

```
INSERT INTO xforms
VALUES(xmltype(getDocument('’personalinfo.xml')),'personalinfo.xml','1','2327','none','
0','none','date()')
```

Figure 9: Inserting XML Type Document

The following are the tables created in this project using Oracle Database to store the information regarding the forms and the users.

XForms: form, formn, version, id, root1, descendants, root2, sdate. This table is used to store XMLType Document. The getDocument() function shown in the above figure is used to convert the XML Document into XMLType Instance. This table contains all the information regarding the form – when it was created, what version is it and is it a branch from an existing form, describes the form etc.

Usertable: name, username, password, id, url – This table stores all the information regarding the user. What is the name of the user, id, password and username to logon, what is the page he last logged on, and security information.

Elepos: name, left, top, id, version, fname. -- This table is used to record all the positions of the elements into the database. These positions are used to design the form.

2.6 DOM Model and the SAX Parser

SAX Parser: A "*Simple API for XML*" [6] that helps in processing XML documents. The SAX Parser reads through the entire file to extract the required tags ex: input, selectOne, textarea. The main advantage of SAX Parser is it is very simple to use. The SAX Parser is an "*event-driven*" interface in which when an "*event*" like recognizing the XML tags or processing an XML document occurs, the parser uses the appropriate methods given to it by the user.

```
org.xml.sax.InputSource
org.xml.sax.SAXException
org.xml.sax.SAXParseException
org.xml.sax.*
```

DOM – Document Object Model

DOM is a platform-independent interface independent of language that is used for accessing XML documents and HTML documents. With the help of the DOM we can access XML documents i.e., we can add new elements, delete elements or modify the structure of the XML document. The main objective of XML DOM is to create a standard programming interface for different applications.

In the code below first the XML Document is passed through the Parser, once the document is loaded the DOM is used to extract the information from it, to navigate around the structure, to add or delete or modify elements.

```
InputSource    input = Resolver.createInputSource (new File (formname));
XmlDocument doc = XmlDocument.createXmlDocument (input, false);
doc.getDocumentElement ().normalize ();
```

Figure 10: DOM Statements

As shown in the above code, the DocumentElement is the root of the XML Document. It organizes the XML document in a tree view. First the root element is present then the child nodes are added. In the example code given below, getChildNodes() method is used to retrieve the child nodes associated with the root element. Also a "Node Interface Model" is used to access the nodes in the Document. Example: NodeList gives the number of elements by the given tag, getAttributes() extracts the text associated with that particular node. Even the "Element model" can be used to access the nodes, in this project we used Element model as we can see in the below example

```
NodeList listOftextf = doc.getElementsByTagName("xform:input");
int totallist = listOftextf.getLength();
Element node3 = (Element) node2.item(j).getChildNodes();
String c2 = node3.getAttributes().toString();
```

Figure 11: Elements

All the above examples explain methods for retrieving XML Document from Oracle XML Database. In the next example we can see methods to add or modify elements of the XML Document. “createElement(“xform:input”) creates an xform:input tag in the XML document. The xform:input tag is used to represent textfields in XForms. “setAttribute()” sets the values of the textfield. “createtextNode() creates the text attribute associated with the tag.

```
Element child = (Element)doc1.createElement("xform:input");
root.appendChild(child);
child.setAttribute("value", tname[i]);
Text child1 = (Text)doc1.createTextNode(tname[i]);
child.appendChild(child1);
```

Figure 11: Creating xform type elements

Chapter 3

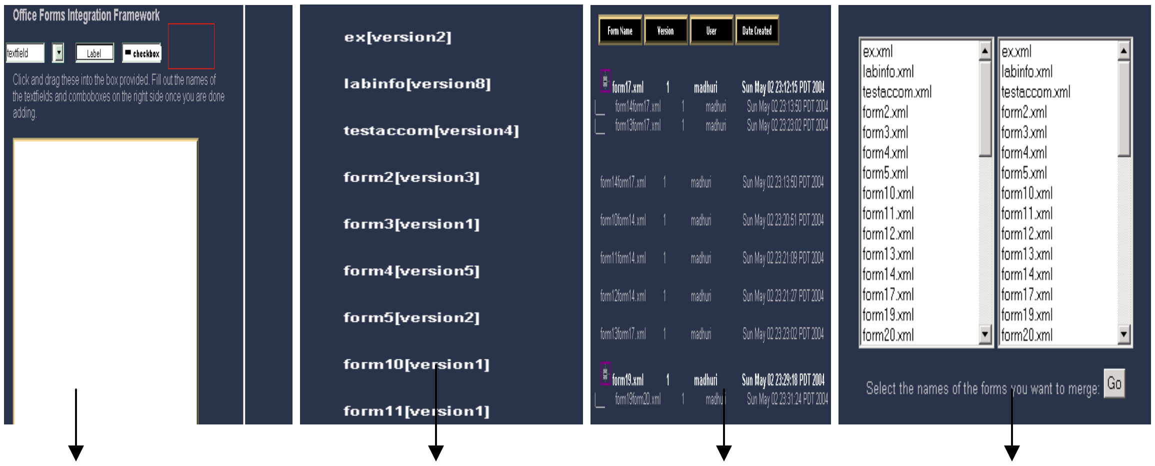
Design and Implementation

3.1 Purpose

The main purpose of this project is to implement a framework that can create, design, edit and merge forms. It should be compatible for integration into other applications. The framework should be able to maintain the revision history i.e., whenever a form is edited, a new version of the form is created and stored in the database. The forms are created using XForms and Oracle database is used to store them. The data in the XForms uses XML to describe the data and XSLT (stylesheet) to display the form. An example of how the framework works is given below. A detailed description of how to create, design, edit and merge forms is described in the following section.

Example:

As you can see in the below figure the framework consists of five parts. Create, design, edit, merge and the Revision Control System.



Click and drag these into the box provided. Fill out the names of the textfields and comboboxes on the right side once you are done adding.

ex[version2]
labinfo[version8]
testaccom[version4]
form2[version3]
form3[version1]
form4[version5]
form5[version2]
form10[version1]
form11[version1]

Form Name	Version	User	Date Created
form17.xml	1	madhuri	Sun May 02 23:12:15 PDT 2004
form14form17.xml	1	madhuri	Sun May 02 23:13:50 PDT 2004
form13form17.xml	1	madhuri	Sun May 02 23:23:02 PDT 2004
form14form17.xml	1	madhuri	Sun May 02 23:13:50 PDT 2004
form10form14.xml	1	madhuri	Sun May 02 23:20:51 PDT 2004
form11form14.xml	1	madhuri	Sun May 02 23:21:09 PDT 2004
form12form14.xml	1	madhuri	Sun May 02 23:21:27 PDT 2004
form13form17.xml	1	madhuri	Sun May 02 23:23:02 PDT 2004
form19.xml	1	madhuri	Sun May 02 23:29:18 PDT 2004
form19form20.xml	1	madhuri	Sun May 02 23:31:24 PDT 2004

ex.xml	ex.xml
labinfo.xml	labinfo.xml
testaccom.xml	testaccom.xml
form2.xml	form2.xml
form3.xml	form3.xml
form4.xml	form4.xml
form5.xml	form5.xml
form10.xml	form10.xml
form11.xml	form11.xml
form12.xml	form12.xml
form13.xml	form13.xml
form14.xml	form14.xml
form17.xml	form17.xml
form19.xml	form19.xml
form20.xml	form20.xml

Select the names of the forms you want to merge:

create forms edit forms version control system merge forms

The above figure shows how the framework looks when you click on each application. In “create forms”; the user can create a form with textfields, labels, and checkboxes. “Edit Forms” screen contains a list of all the forms in the database. The latest version of the form is provided. In the version control system all the forms and their revisions are displayed. Even the date and time of creation is displayed. The last screen lets users merge two different forms. A description of the process is provided in the Implementation Section.

3.2 Design

This section of the report discusses the different types of object-oriented concepts used in this project. Usability and security are some of the important concepts when developing applications such as this project. The quality of the project has to be maintained, for this purpose usability patterns have been implemented in this project. Application Security is another important concept, there are many patterns that can be implemented in it. These patterns will be discussed in the following sections.

3.2.1 Usability patterns

The quality of a framework depends entirely on the usability of the framework. It should be user friendly and simple to understand. A collection of Usability patterns has been discussed in www.designforquality.com. The following are some of the Usability patterns that has been implemented in this project to improve the quality of usability property:

History Logging: It would be very helpful to log all the actions of the user such as the file last used, or version of the file created. This pattern is implemented in this project in the Version Control System (VCS). The VCS records all the information regarding the forms pertaining to a particular user. It creates a history log listing all the forms created, the date and time they were created, the version of the form, the date it was last modified, and description of the changes made to the form. This type of pattern was implemented in the project because it would be very helpful to keep track of actions of the user so that the user can get back to the log and check to see what went wrong if an error occurs.

System Feedback: If a user makes a request that is not working, then the system should be able to send alerts stating the problems. For example when we try to download a document the system alerts the user by sending "Do you want to open or save the document?" These system feedbacks help the user in being well informed about the system state. In our framework when the user tries to enter wrong data the system pops up an alert stating “wrong information”. This pattern prevents the user from performing actions that might lead to errors.

User Modes: Each user has their preferences. They should be able to design according their own preferences. In this project the user can design their own form that is the user is able to drag the elements in the form according to his/her preferences. This lets the user personalize forms according to his/her needs.

Workflow Model: The last of the usability patterns implemented in this project. In this model the user has to follow specific steps in order to complete the project. Example: Content Management System. In this project the user has to be first registered to logon. This is the first step in the process. Once the user is logged on they need to create a form, then they can later edit it or merge it. Once the form is done it can be rearranged/designed according to the user’s needs.

3.2.2 Application Security

A list of patterns has been described by Joseph Yoder [8] in “Architectural patterns for enabling application Security” to implement the application security. The following are the patterns that have been implemented in this project:

Single Access Point:

If we have more than one access points to enter the application it would be difficult to keep the application secure. In this project we have different applications that need to be accessed by a single user. Having different logins for each would not be secure enough. Having one secure single access point solves this problem. We have four different operations that need to be done create forms, edit, merge and design. Everything is combined and setup in such a way that the user can access all the applications by providing username and password at a single access point. All the user information that needs to be used in the entire application is collected at that single access point. Single administrator account was created so that all the information can be accessed and checked by the administrator. All the information about all the users can be viewed in this account.

Check Point

Single Access Point helps in providing a single gateway for the entire application; the entire user information that is needed is collected at this single access point. But this is not enough, as it cannot protect the system from hackers. We can prevent the hackers by implementing security measures like disabling the account if a wrong password is entered. But, if a real user forgets his/her password it would be very difficult for him/her to enter the application. For this purpose Check Points are used so that a user can enter wrong password only a certain number of times. The flow diagram given below describes how checkpoint is implemented in this project.

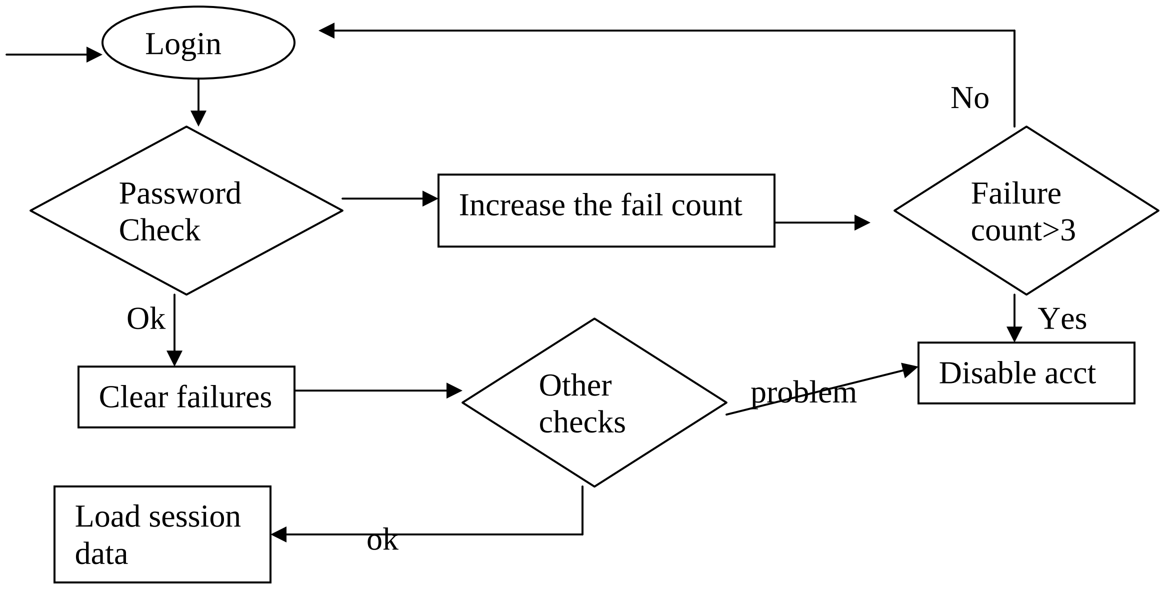


Figure 12: CheckPoint implementation

Sessions

A session object can be created at the time of login so that the information that needs to be used in all the applications can be stored in a session object. In this project once the user has logged on, the user information is stored in a single session object so that it can be shared and accessed by all the applications.

3.3 Implementation

Office Forms Integration Framework:

The main objective of this project is to create a framework that lets any user create, design and edit forms. It should also identify forms associated with a person in the database.

Below is an overview of how the framework works:

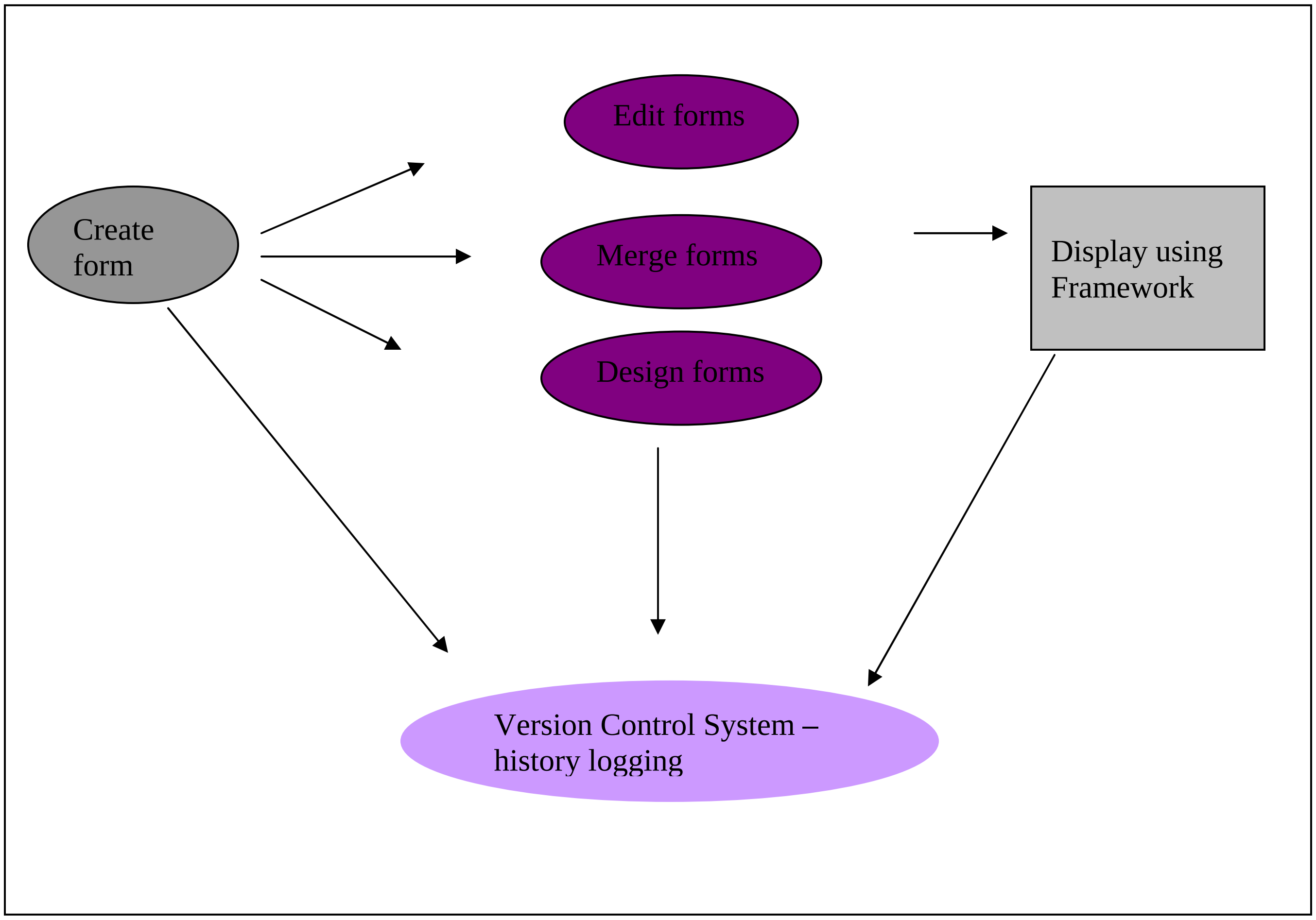


Figure 13: Overview of Office Forms Integration Framework

3.3.1 Version Control System:

Need for Version Control System:

As technology advances, the process of storing information on paper is being eliminated. This creates a need for backing up information and data on applications. Storage Tapes are used to backup all the information. But this is not a very effective way of storing all the revisions; these devices overwrite the existing file thus not letting the user go back to a particular version of the file. An alternative to these approaches would be to implement version control systems that would never delete the file or replace a file; a new version is created whenever a change is made.

Version control system (VCS) needs to keep track of what’s going on in the entire application and organize the different versions of the forms existing in the system. Since the forms can be edited any number of times we need to keep track of all the changes being made. The VCS was developed to maintain the revision history and style of the form. All the actions of the user are logged. It even keeps track of the last page accessed by the user. Each time the user enters a page the VCS is updated with the latest userdata so that it will help the user keep track of what he/she was doing their last login. The VCS is organized in the form of a tree, i.e., the root would be the original form, the child nodes being the revisions made to the form and also branches emerging from the child nodes. This VCS is implemented for individual users. So the problem of two users using the same file does not occur. For this reason-locking feature has not been implemented.

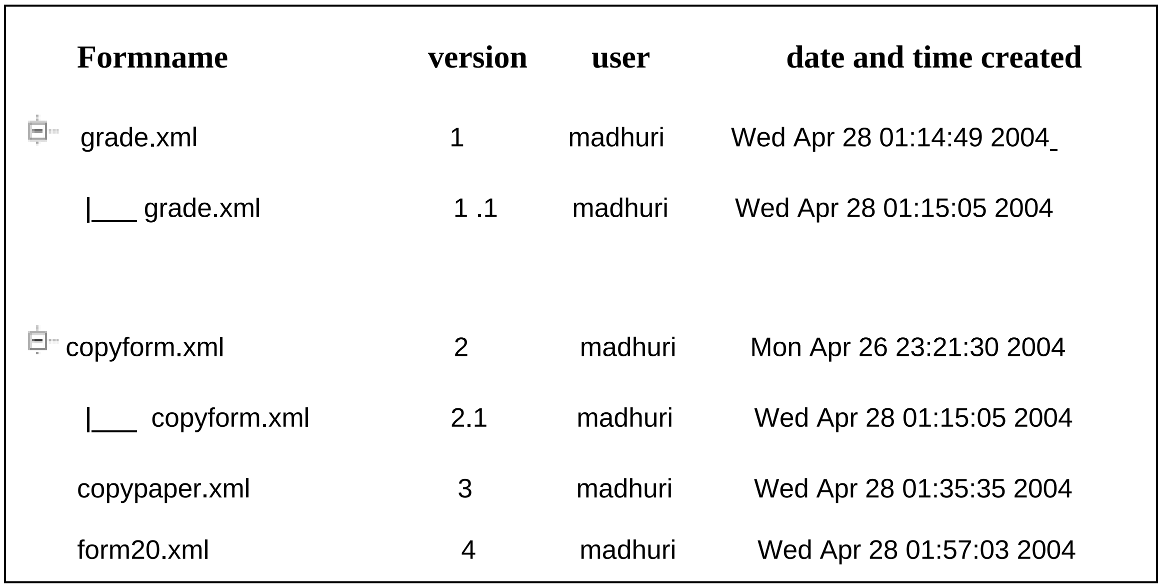
The Version Control System should have the following features:

(a) Ability to create new branches of existing forms as well as to create new forms.
(b) Ability to merge existing branches and to unify form data from two branches.
(c) Ability to examine change histories along a given form branch.

Whenever a new form is created all the information regarding the file is stored and it is given a version number according to the order in which it was created. Once the user edits it, the VCS saves a new version of the form, the date and time the change was made and description of the changes made. The numbering starts at 1.1 the first time a change is made. The 1 before the dot represents the original version of the form. The revisions can be specified in two different ways: display based on the version or the date and time it was created.

Another ability of the application is to create new forms from the existing branch or merging them. Branches can be created from a new form or from a version of an existing form. In the case of merging the system checks for the recent version of the forms and then merges them. In the framework there are five buttons for different purposes: create new forms, edit forms, design forms, merge forms and the last function shows the information stored in the VCS. In here the name of form, version, date and time it was created are displayed. The forms displayed can also be sorted according to the form name by clicking on the “Form Name” button. The VCS checks all the versions of the forms and decides which version to let the user use. In the case of editing forms only the latest version of the form is provided, changes are made to the latest version. The user can go back and look at the VCS to look at all the versions. There is also an option for the user to enter the version of the form they want to look at.

Version Control System:



Formname	version	user	date and time created
⊟ grade.xml	1	madhuri	Wed Apr 28 01:14:49 2004_
|___ grade.xml	1 .1	madhuri	Wed Apr 28 01:15:05 2004
⊟ copyform.xml	2	madhuri	Mon Apr 26 23:21:30 2004
|___ copyform.xml	2.1	madhuri	Wed Apr 28 01:15:05 2004
copypaper.xml	3	madhuri	Wed Apr 28 01:35:35 2004
form20.xml	4	madhuri	Wed Apr 28 01:57:03 2004

Figure 14: Display of forms and their versions

Existing Version Control Systems:

Different techniques are used in implementing Version Control Systems. Let us take a look at two different Version Control Systems and compare them to our Version Control System.

(i) *Revision Control System (RCS) [9]*

(ii) *Concurrent Versions System (CVS) [10]*

RCS was developed by Walter Tichy at Purdue University; widely used by Unix Systems. RCS lets people create different versions of the text, i.e., if an error occurs the user can check at a previous version of the text. If a user is working on a text and another user wants to work on the same file, RCS does not let the user open the file until the first user is done. On the other hand CVS is also used to maintain a history of all the changes made to a set of file. But CVS lets two users open the same file at the same time. It lets both users make changes and once they are done merge both changes into the document. But if the changes are made to the same line of text then none of the changes are saved. The user has to manually make the changes. The Version Control System developed by this application also keeps track of all the changes made to the forms and also follows a tree view to store the versions. As this system is used one user at a time no locking mechanisms are implemented on this system. CVS is more complex than RCS and needs more disk space. But it is very useful for applications that are at different places and are connected by the Internet. Even the Version Control System developed by this project can be used by applications that are distributed at different sites.

Let us a take a look at all the actions performed in the Framework:

3.3.2 Create New Forms

Once you login into the system, the first thing to do is to create new forms. The user can create textfields, checkboxes, comboboxes, labels, textarea. We can drag the icons into the box provided. Once the user drags the required number of textfield, checkboxes the information about them is filled out on the right side. These elements are dragged and counted by the following functions. “document.onmousemove = mousemove” enters the “mousemove” function whenever the mouse moves. Once the user clicks on the icon it enters the “Mousedown” function.

In “Mousedown” function the actual element is cloned to create another element (textfield or checkbox etc) and inserted in the document. Also the number of elements added is counted. Each time the mouse enters the “mousemove” function the left and top positions are being calculated to keep track of the location of the element. Once the element is placed in the desired location it enters the “mouseup” function. In here the final positions are calculated. This is how the elements are dragged and calculated; a summarized version of the three functions is described below:

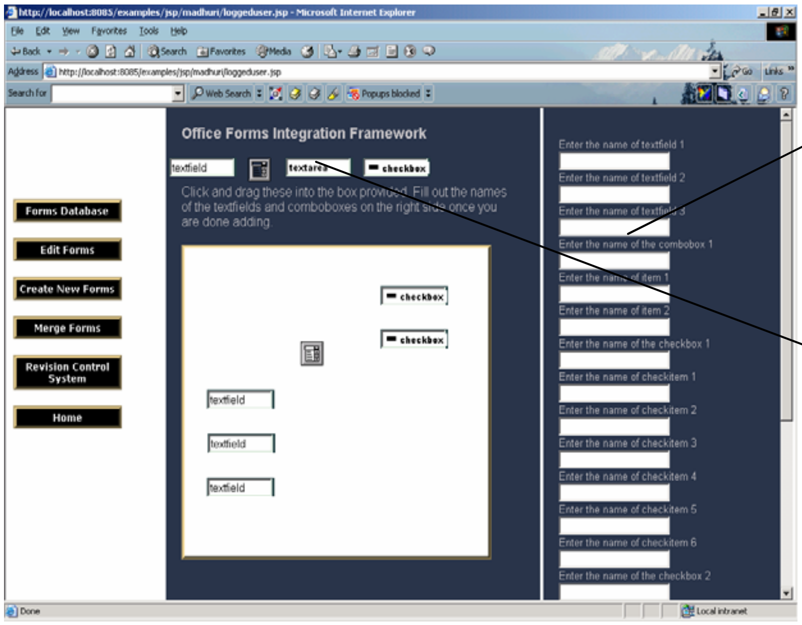
```
function mousedown(e)
{
        doDrag=true
        -----------
 var oCloneNode = o.cloneNode(true)              }  ------>   cloning original element and inserting cloned element
 document.body.insertBefore(oCloneNode)          }             in the document
 document.form2.count1.value=countt;
 document.form2.count2.value=countc;
 document.form2.count3.value=countta;
 document.form2.submit()
}

function mouseup(e)
```

```
{
        doDrag=false
        oLeft = e.clientX-mouseLeft
        oTop = e.clientY-mouseTop
        --------
}
function mousemove(e)
{

        if (doDrag)
        {
                oLeft = e.clientX-mouseLeft
                oTop = e.clientY-mouseTop          sets the positions of the elements
                setPosition(o,oLeft,oTop)
                return false
        }
}
```

It can be shown in the snapshot provided below:


http://localhost:8085/examples/jsp/madhuri/loggeduser.jsp - Microsoft Internet Explorer
Address http://localhost:8085/examples/jsp/madhuri/loggeduser.jsp
Office Forms Integration Framework
textfield
textarea
checkbox
Forms Database
Edit Forms
Create New Forms
Merge Forms
Revision Control System
Home
checkbox
checkbox
textfield
textfield
textfield
Done
Local intranet

Click and drag these into the box provided. Fill out the names of the textfields and comboboxes on the right side once you are done adding.

Enter the name of textfield 1
Enter the name of textfield 2
Enter the name of textfield 3
Enter the name of the combobox 1
Enter the name of item 1
Enter the name of item 2
Enter the name of the checkbox 1
Enter the name of checkitem 1
Enter the name of checkitem 2
Enter the name of checkitem 3
Enter the name of checkitem 4
Enter the name of checkitem 5
Enter the name of checkitem 6
Enter the name of the checkbox 2

This is where all the names of the tags are entered.

These icons for textfields, checkboxes --- are dragged into the box provided below

Figure 15: Snapshot of Create Forms

Once the information is entered it is passed to another form so that an XForm document can be created. In here we use the DOM and the SAX Parser to create an Xform document. A brief description how the form is created is given below. The information regarding the number of elements to be created and the names of the elements is collected and created into XForm. As you can see below root of the element is created first and then the other elements are attached to the root. The example provided below shows about creating “xform:input” tag. All the other elements are created in the same manner and attached to the root.

```
XmlDocument doc1 = new XmlDocument ();
Element          root = (Element) doc1.createElement ("html");
Element child = (Element)doc1.createElement("xform:input");
root.appendChild(child);
```

The layout of the form has to be consistent with the format specified. The ‘layout tag’ has the style information and the ‘formdocument’ contains the Xform document. The following are the steps followed in this part:

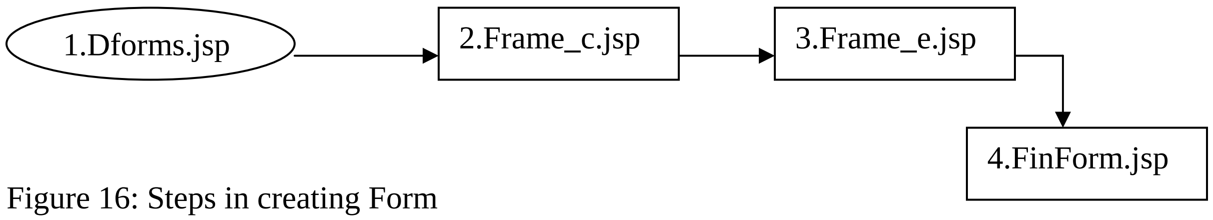


Figure 16: Steps in creating Form

In the first step two frames are displayed, one empty and one with a number of icons representing textfields, checkboxes, etc. to be dragged into the box provided.

In the second step once the icons are dragged the total number of each type of element is taken into account and also the information entered for each element is passed to Frame_e.jsp (3rd step).

In the last and final step FinForm.jsp the XForms document is created.

3.3.3 Edit Forms:

Editing forms is the most important part of the Framework. Once the user creates a form he should be able to make modifications to it. At the same time the version control system discussed in the following sections starts saving the information regarding the user, actions performed and many more. Editing Forms has five operations associated with it.

i. Delete Items

ii. Add new element

iii. Rearrange Elements

iv. Reset to original form and

v. Finished Editing, Save Form.

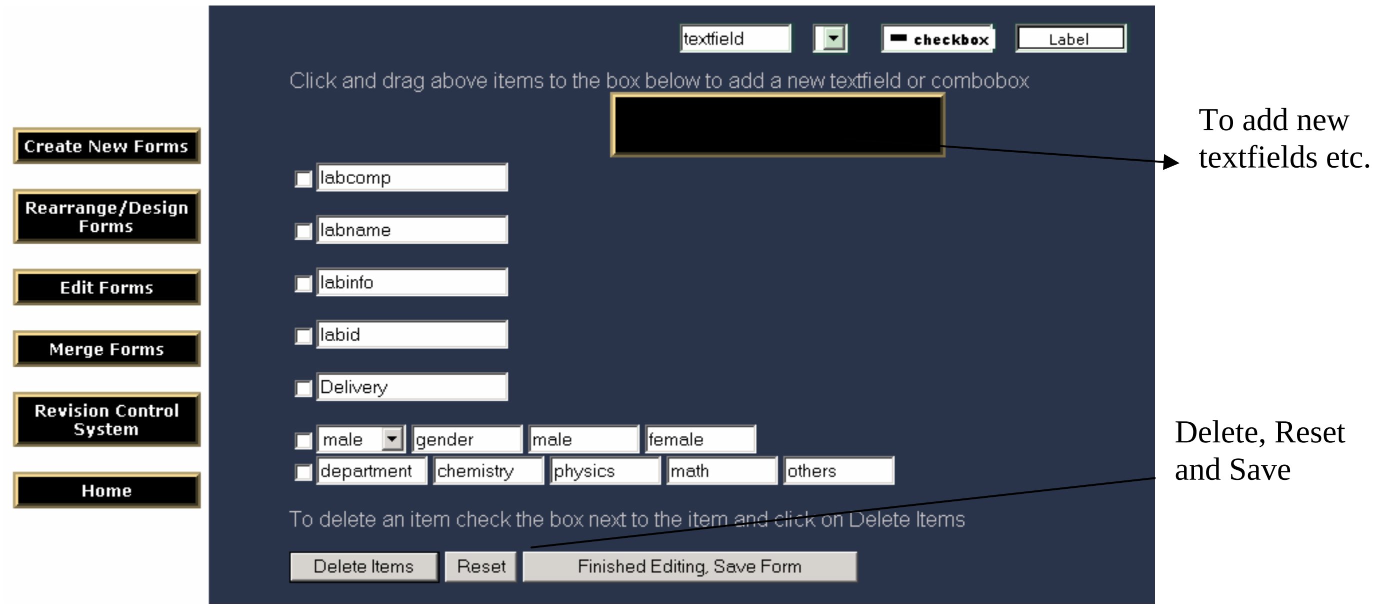


Figure 17: Snapshot of Edit Forms

“Delete Items” As you can see each element has a check box associated with it. The user can check it and click on the “Delete Items” to delete the items. If a user accidentally deletes an element it can be “Reset” to the original form by clicking on the “Reset” button. A list of icons – textfields, checkboxes etc is provided (just like – create forms) to add new elements to the form. Click and drag the icons to the box provided so that they can be added to the Form.

Once all the actions that need to be performed are done, click on the “Finished Editing, save Form” to save the form. Once this button is clicked a new version of the form under the same form name is created and inserted into the database do that the VCS can keep track of all the actions of the user.

3.3.4 Merge Forms

Sometimes a user is required to create a form that has information already created is contained in two different forms. In such a case the user can merge these forms to get the required form. This helps in reducing redundancy. If they were paper forms the user has to recreate the form, but “merge forms” application just merges them and creates the required form in an instant. We can also add more elements/ rearrange them in the newly created merged form

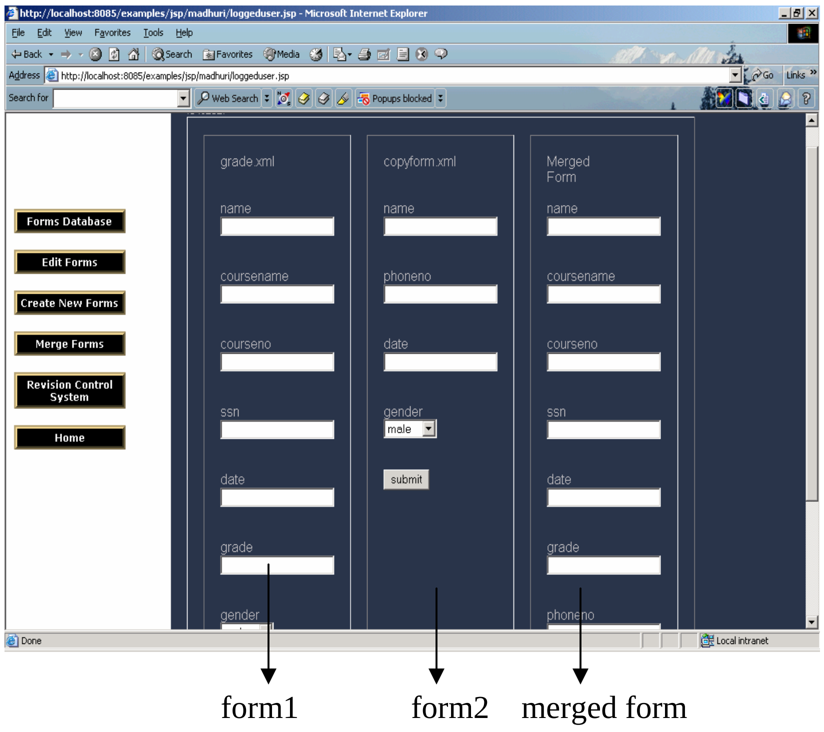


Figure 18: Snapshot of Merge Forms

3.3.5 Design Forms

In this method the user can move the elements around to define the positions of the elements. The left and the top position of each element are stored in the layout of the Xform document. Once the elements are positioned it is then used by the XSLT stylesheet to transform the Xform document and display it.

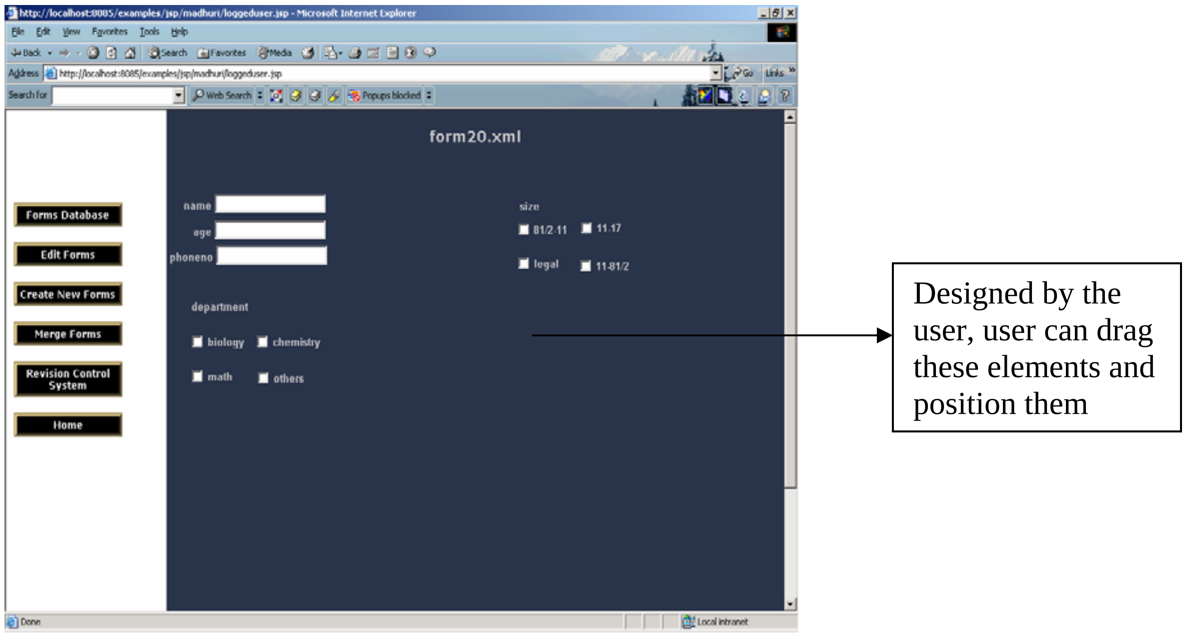


Figure: 19 Snapshot of Positioning elements

Version Control System:

In brief, VCS logs the information about the user and actions performed by the user from the minute the user registers with the system. When a new form is created, the name of the form as well as the form itself is stored in the database to keep track of the information that has been changed. The version number is also stored to identify different versions of the form. If a user edits the form, the VCS notes which elements are added, modified or deleted so that they can be used for future reference. When the user is in the Merge operation, the data collected by the VCS helps the user by not letting them merge the forms that were already merged. All the positions of the elements are also updated immediately after a change is made. Only the latest positions of the elements are stored in the database.

All the steps followed in the Framework:

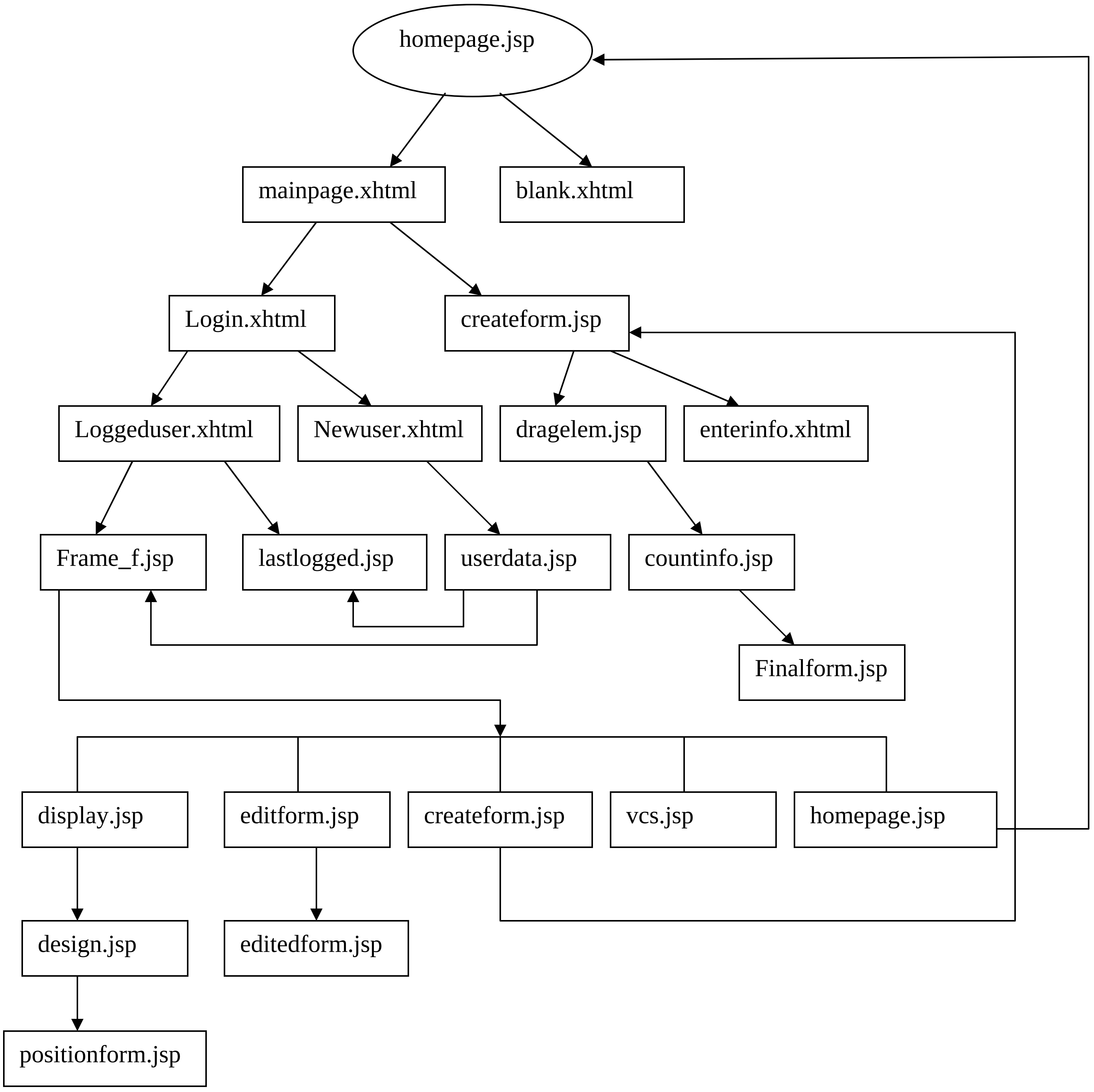


Figure 20: Framework

Chapter 4

Usability Testing

Usability testing is a means of finding the usability of a product. Making a product usable can mean many things such as making it easy to use, error free, simple to navigate etc. The main purpose of Usability Testing is to see if the user is satisfied with the product and that all the needs of the user are met. Many important concepts for designing simple web sites have been discussed in Jakob Nielson's "Designing Web usability: The Practice of Simplicity". To test the usability of the Framework developed in this project five different test subjects were selected. These five users were selected from a diverse group of individuals with varying degrees of computer experience. All of them had used forms in their daily life. Prior to the test these users were given a brief tutorial on the purpose and usage of the framework. The following were the main criteria in observing the users:

1. Were the users able to figure out how to use the framework on their own?
2. What were the common mistakes made?
3. How long does each user take to learn about the framework?
4. Is the user satisfied with the product?
5. What were the recommendations made to improve it?

Each user was tested on a specific set of tasks, these included:

- A paper-based form was given and they were asked to create a similar form using the program.
- They were asked to merge two forms to create a new form
- An existing form was to be edited to create a new version
- General navigation around the various applications

Results:

The following are the results of the Usability Testing:

1. One of the users suggested providing instructions before each section on how the section works and how to use it. The first time the user logged into the framework without any instructions he was overwhelmed and did not know how to proceed. The user found easier to navigate after the instructions were given.
2. At first the form fields only included textboxes, comboboxes and checkboxes. Based on user feedback labels were later added, as the users felt it would be more useful to have comments in the forms.
3. Font size and font color of the forms were predefined in the Program. The users felt that they would like to have more control in designing the forms. The latest version of the program now allows users to customize the font colors, sizes etc.

4. In the ‘edit forms’ section adding new fields to the forms was implemented. While testing this section one of the users inadvertently added a checkbox and wanted to remove it, but there was no way to delete fields. I have now changed the ‘edit forms’ so that the user can delete any fields from the form. The form can also be reset to the original version.
5. The Version Control System displays all the forms created in the database and the date they were created. One of the users suggested it would be nice if it also showed a description of the changes made when a new version of the form is created. A description field was added to the VCS system for this purpose.

	Create new Forms	Design Forms	Edit Forms	Merge Forms	VCS	Time Taken
User 1	Was totally confused didn’t know where to start	Had no problems	Tried to remove the field after adding a textfield	Tried to select more than 2 forms	Liked the idea that she can see all the versions and date they were created	45 minutes
User 2	Started without any problems. Tried to move already added field	Didn’t know how to move around, needed instructions	No problems	No problems	No problems	60 minutes
User 3	Had no problems	Liked the idea of moving around the fields and positioning them	No problems	No problems	Suggested to add a description field so that the user would know what changes were made	30 minutes
User 4	Didn’t know where to start, had problem dragging the fields	No problem	Needed more instructions	No problem	No problem	40 minutes
User 5	Suggested to provide instruction before each section	No problems	No problem	No problem	No problem	50 minutes

Figure 21: Usability Testing Results

Average time taken by the users --- 45 minutes

Most of the test subjects found the Framework to be very useful. They liked the idea of being able to not only design and create their own forms but also go back and edit them later. Each time the user made changes a new version was created and the changes made to the previous version were stored. The Version Control System has an ancestral view in which all the forms that were created in the database are show. Users were pretty impressed with the Framework; they felt that it would be a good tool in implementing forms if the data was also stored in the database.

Chapter 5

Conclusion

As we can see from the above discussions “Forms” are the most efficient way of gathering and processing information. They enable the transactions on the web to communicate with the user in a simple and efficient manner. XML has become the ideal language for “Automated Processes” and “Integration” because it can be transformed into any format depending on the need of the user. Different users can receive the same XML file and use it according to their specific needs. XForms are the future of HTML forms; they use XML to describe the data and HTML to display the forms. This separates the data from the presentation; making them usable on any platform. In this project we created forms using XForms with the style information described in them. Once the XForms are created XSLT stylesheets were used to transform the XForms and display them.

Merging XML data with Relational Data is also becoming essential because the databases can, not only store XML data but can also solve the difficulty in working with XML. Relationships between data can be represented by both XML and Relational databases; combining both would provide more efficient methods. In this project we used Oracle XML DB to store, retrieve, query and merge XML Documents. Structured Method was used to store data as it can help to retain the hierarchy of the XML Document. This helps to use relational queries of merging and manipulating the XML Documents.

Version Control System was implemented to keep track of all the actions of the users and organize the different versions of the forms. VCS uses ancestral tree view to organize the different version of the forms. It was developed to maintain the revision history and style of the forms. The following are the features of the VCS

(a) Ability to create new branches of existing forms as well as to create new forms.
(b) Ability to merge existing branches and to unify form data from two branches.
(c) Ability to examine change histories along a given form branch.

Two object oriented concepts were used in the design of the Framework: Usability patterns and Application Security. The quality of a framework depends entirely on its usability. It should be user friendly and should not be too complicated for the user. It should also be secure enough to prevent hackers. Various Security issues have been implemented, such as allowing the user only three consecutive attempts to login into the system, creating sessions so that user information is collected only once and transmitted whenever needed.

This Framework lets the user create XForms, edit and merge forms without knowing the underlying technology. In general if a user needs a particular type of form, someone else has to design the form before they can use it. But in this system every user can design and customize their forms. The revision history of the forms is also created so that if an error occurs at some point in the creation they can go back to the previous version. As everything is being stored in the database it reduces disk space making it more efficient. We use XML to describe the data, XSLT to transcribe it and Oracle to store it. Because of this advantage XML lets us work on any platform. By using Oracle XML DB to store the data we were able to preserve the hierarchical information of the XML document. This also enabled us to query, modify and manipulate the XML Data. All this technology has helped us to generate a user friendly framework that lets users create, edit, merge, design forms maintain revision history. Sequences of test forms have been created to test if the Version Control System is functioning properly. Furthermore, simple usability testing has been carried out.

Future Enhancements

In this project we are able to store the forms created in a database. The data entered into the forms is not being stored in the Database. Storing the data would make the Framework a toolkit for XForms. A future improvement to the framework could be that when a form that has already been created is opened, the information pertaining to it can be retrieved so that the user need not re-enter it.

The Framework can be made to be more accessible for any type of user. Accessibility issues such as providing alt tags could be implemented.

References

1. Harold R.E.,& Means S.W. XML in a Nutshell (2nd ed.). O'Reilly. 2003.

2. Sperberg-McQueen C.M., & Thompson Henry. XML Schema. W3C. http://www.w3.org/XML/Schema. 2003.

3. Thierry Michel. XForms - The Next Generation of Web Forms. W3C. http://www.w3.org/MarkUp/Forms/. 2003.

4. Oracle9*i* XML Database Developer's Guide - Oracle XML DB.

5. Shimura T., Yoshikawa M., Uemura M. Storage and Retrieval of XML Documents using Object-Relational Databases. Proceedings DEXA Conference Florence, Italy, 1999. pp 206--217.

6. Simple API for XML http://sax.sourceforge.net/

7. Article on patterns. www.designforquality.com

8. Yoder Joseph, Barcalow Jeffrey. Architectural Patterns for Enabling Application Security 1998

9. Tichy W. Design, Implementation, and Evaluation of a Revision Control System, Proceedings: 6th International Conference on Software Engineering pp. 58-67. IEEE Computer Society Press, 1982

10. Concurrent Versions System (CVS) www.cvshome.org

11. Tutorials on XHTML, XForms, XML, DOM. www.w3schools.com

12. Example on importance of XML. www.congressonlineproject.org/glossary.htm

13. Suciu D. On Database Theory and XML.SIGMOD Record (8), pp.39--45. 2001.

14. Su H., Kramer D., Chen L., Claypool K., and Rundensteiner E.A. XEM: Managing the evolution of XML Documents. Eleventh International Workshop on Research Issues in Data Engineering (RIDE 2001). Heidelberg, Germany, April 1-2, 2001.

15. Tatarinov I., Viglas D.S., Beyer K., Shanmugasundaram J., Shekita E., Zhang C. (2002,June). Storing and Querying Ordered XML using a Relational Database System. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pp. 204--215.

16. Database management. www.infoworld.com/techindex/xml.html