Transformation of XAML schema for RIA using XSLT & UsiXML

Francisco J. Martínez Ruiz [1], Jaime Muñoz Arteaga [2] and Jean Vanderdonckt [3].

[1] Universidad Autónoma de Zacatecas. Departamento de Ingeniería en Computación, Av. López Velarde No. 801, Zacatecas, Zac., 98060. México. jmartinez@uaz.edu.mx
[2] Universidad Autónoma de Aguascalientes. México. jmunozar@correo.uaa.mx
[3] Université Catholique de Louvain. Belgium. vanderdonckt@isys.ucl.ac.be

Abstract. User interface design and development for Rich Internet applications is a difficult task with actual tools. The designers must be aware of the computing platform, the user's characteristics (education, social background, among others) and the environment within users must interact with the application. Several model based technologies have been proposed and in this study we review a XML-compliant User Interface Description language: XAML and we propose a translation schema with XSLT for generate user interface descriptions on UsiXML. This way, we can avoid the dependency to proprietary engines for designing tasks.

Keywords: Rich Internet Applications, User Interface Design, Human-Computer Interaction, Software engineering, Information systems.

1 Introduction

Building a model based application [1] requires a framework to define the design steps needed for describe our computer system, including the features: Multi-level abstraction, Modality independence, among others [2]. The Cameleon Reference framework [3] expresses these features to describe an application. This framework structures the development process within four levels of abstraction: Task and concepts, Abstract User Interface (AUI), Concrete User Interface (CUI) and Final User Interface (FUI). To denote a UI at any level of abstraction, it's required a User Interface Description Language (UIDL) [4]. One of theses description model based languages is UsiXML (UsiXML which stands for User Interface eXtensible Markup Language). This language incorporate the four abstraction levels of Fig. 1 as described in [5]. UsiXML describes the UI for multiple contexts e.g., Character User Interfaces, Graphical and Multimodal ones in a form that maintains design independent from specific platforms [2].

The interest in building a Web Rich Client has been increasing since a couple of years. This User Interface type has similar features of those provided in typical desktop applications, e.g., robustness, better responsiveness and visually more appealing than the classic HTML ones. RIAs technologies help us to reach this goal [6]. RIAs are Web applications that transfer most of the load of processing the user interface to the Web client while the predominant part of data (from control and maintaining to

business data) remains on the application server. A standard RIA architecture (Fig. 1) includes an application controller and an application server that control the Web Services Calls that use a XML dialect to transfer data and layout information. Note: Recent Databases can handle also XML with this; the process of translation is pursuit by XQuery language [7].

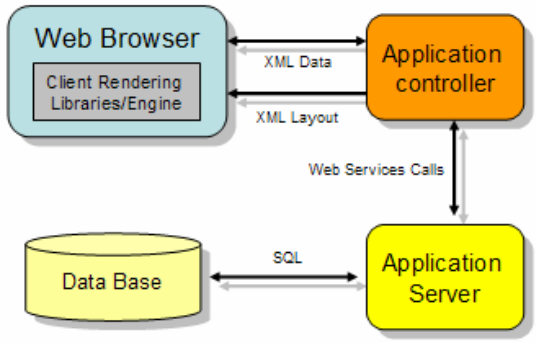

Fig. 1. Typical architecture of a RIA application.

2 Problem description

Our study problem is a well known scenario for developer teams: After embracing a technology if you must build (or migrate) an application but in another technology is mandatory recoding most of the project even with RIAs where as part of the core of these technologies resides a XML-compliant Description Language. These languages have a common objective: model the User Interface, The most popular options to design and develop a RIA application (at FUI level) are: XUL[8] openlaszlo [9], Flash&Flex from MacroMedia [10] and one proposed by Microsoft: XAML [11]. The Windows Standard Development Kit for the new operating system Longhorn, WinFX SDK [12] contains a Representation subsystem called Avalon which integrates XAML. These languages are similar but no interchangeable so there is a problem when is needed "retargeting". The underlying problem is the translation from $language_1$ to $language_2$ many solutions have been proposed in order to support this translation (which can be called with these names UI forward engineering, reverse engineering, and reengineering) e.g., techniques based on graph grammars and graph transformation that produce after a successive series of transformations generates a target UI from a initial one [13]. After the literature review, we recognize as one of the most promising efforts XSLT transformation schema [1, 2, and 4] and we would like to exploit it in our study.

3 Contribution

In this paper we proposed the construction and settling of the rules needed to construct an XSLT specification to transform a source GUI defined in UsiXML in the Concrete User Interface level into a XAML implementation. The proposed schema results in the translation of the UI basic elements: windows, buttons, and textboxes, among others. In order to do so, Xpath expressions [7] were written to get access to the UsiXML nodes that match the XSLT rules of substitution.

This work would be the initial step for modeling RIA applications with UsiXML and second, to provide a platform independent way to design RIA applications without being attached to a proprietary solution, XAML in the near future is going to be one of the most used solutions to create such applications and would be desirable to provide a neutral development language to build RIA User Interfaces that is the goal of this paper. The architecture of this transformation application is shown in figure 2. It's a typical XML to XML transformation schema; in Fig. 2.1 the input document is specified in CUI layer of UsiXML that is a concrete version of the elements defined in the abstract level besides layout and navigation behavior [3].

The widgets are recognizable UI elements still not attached to a particular toolkit. That targeting to a specific toolkit is settled in the FUI (Fig. 2.4) which is the production of UI code to be compiled or interpreted, here in XAML.

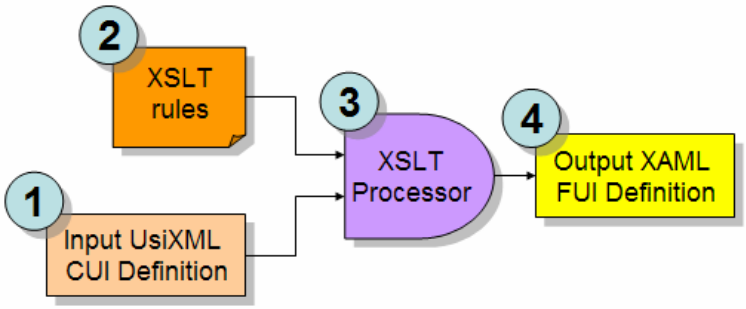

Fig. 2. XSL transformation schema from UsiXML to XAML.

4 Study Case: the poll form

In this section a study case is presented. Our scenario is the design of a minimalist polling system. A basic form available in all of RIA languages [8, 9, 10, 11] composed by a text label, three radio selectors and a submit button. The UsiXML presentation of this UI is shown in Figure 3. The UsiXML document is much more extended and complex than the fragment that is presented which included the resources and events associated to the widgets besides information from the other levels. Some of the saved information in this file would be very useful for retargeting tasks. Because, some data is dismissed by some platforms but it's very important to others e.g., "***groupName***" (Fig. 4) is the name of a container to define a radio button set another example of these

optional features is the "***isVisible***" attribute that would or not be available in our target language.

```
<cuiModel id="poll_system-cui_30" name="poll system-cui">
<window id="window_component_0" name="window_component_0">
<box id="box_1" name="box_1" type="vertical">
<radioButton id="radiobutton_component_2" name="radiobutton_component_2"
content="/uiModel/resourceModel/cioRef[@cioId='radiobutton_component_2']/resource/@content"
        defaultContent="Isaac ASimov" isVisible="true"
        isEnabled="true" textColor="#000000" groupName="asking"/>
<radioButton id="radiobutton_component_3"name="radiobutton_component_3"
content="/uiModel/resourceModel/cioRef[@cioId='radiobutton_component_3']/resource/@content"
        defaultContent="H.G. Wells" isVisible="true"
        isEnabled="true" textColor="#000000" groupName="asking"/>
<radioButton id="radiobutton_component_4" name="radiobutton_component_4"
content="/uiModel/resourceModel/cioRef[@cioId='radiobutton_component_4']/resource/@content"
        defaultContent="Arthur C. Clark" isVisible="true"
        isEnabled="true" textColor="#000000" groupName="asking"/>
<button id="button_component_5" name="button_component_5"
content="/uiModel/resourceModel/cioRef[@cioId='button_component_5']/resource/@content"
        defaultContent="send" isVisible="true"
        isEnabled="true" textColor="#000000"/>
<outputText id="output_text_component_6" name="output_text_component_6"
content="/uiModel/resourceModel/cioRef[@cioId='output_text_component_6']/resource/@content"
        defaultContent="What sci-fi author is your favorite?"
        isVisible="true" isEnabled="true" isBold="true" textColor="#000000"/>
    </box>
    </window></cuiModel>
```

Fig. 3. Polling System UsiXML input.

4.1 Conversion to XAML

This section describes the way XSL transformations are applied to generate XAML output. We write an XSL style sheet which can be used employing a XSLT processor like <oXygen/> XML Editor[14]. This turns into XAML, our UsiXML input document.

This is an excerpt of the final version of the XSLT template rules (Fig. 2) we just add here the needed rules to process our study case since the XML source document is very different to the final document, some of the code is restricted to default values (specially, values that normally are defined by programmers at design time for instance, widget positions).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:wf="http://schemas.microsoft.com/2003/xaml/" version="1.0">
  <xsl:output method="xml" omit-xml-declaration="yes" />
  <xsl:template match="*|/">
    <wf:Form Text="Form1" Name="Form1" ClientSize="400, 400"
      xmlns="http://schemas.microsoft.com/2003/xaml/" xmlns:def="Definition" xmlns:wf="wf"
      def:Class="XamlonApplication1.Form1" def:CodeBehind="Form1.xaml.vb">
      <xsl:apply-templates select="/cuiModel/window"/>
    </wf:Form>
```

```
  </xsl:template>
  <xsl:template match="window">
    <wf:Form.Controls>
      <xsl:apply-templates select="/cuiModel/window/box/radioButton"/>
      <xsl:apply-templates select="/cuiModel/window/box/button"/>
      <xsl:apply-templates select="/cuiModel/window/box/outputText"/>
    </wf:Form.Controls>
  </xsl:template>
  <xsl:template match="radioButton">
    <wf:RadioButton Text="{@defaultContent}" TabIndex="3" Name="{@name}"/>
  </xsl:template>
  <xsl:template match="button">
    <wf:Button Text="{@defaultContent}" TabIndex="2" Name="{@name}"/>
  </xsl:template>
  <xsl:template match="outputText ">
    <wf:Label Text="{@defaultContent}" TabIndex="4" Name="{@name}"/>
  </xsl:template>
</xsl:stylesheet>
```

Fig. 4. XSLT document generated for processing our Poll System.

The resulting XAML UI definition is shown below. The UI definition in UsiXML documents describe the event response (code included in a separated section). This is also the case of XAML that in a separated document, denoted by the "***CodeBehind***" tag includes this information. Also, the size of the widgets is omitted for the sake of simplicity along with the problem described in section 5.2.

```
<wf:Form xmlns:wf="wf" xmlns="http://schemas.microsoft.com/2003/xaml/" xmlns:def="Definition"
Text="Form1" Name="Form1" ClientSize="400, 400" def:Class="XamlonApplication1.Form1"
def:CodeBehind="Form1.xaml.vb">
        <wf:Form.Controls xmlns:wf="http://schemas.microsoft.com/2003/xaml/">
        <wf:RadioButton Text="Isaac ASimov" TabIndex="3" Name="radiobutton_component_2"/>
        <wf:RadioButton Text="H.G. Wells" TabIndex="3" Name="radiobutton_component_3"/>
        <wf:RadioButton Text="Arthur C. Clark" TabIndex="3" Name="radiobutton_component_4"/>
        <wf:Button Text="send" TabIndex="2" Name="button_component_5"/>
        <wf:Label Text="What sci-fi author is ypur favorite?" TabIndex="4"
        Name="output_text_component_6"/></wf:Form.Controls>
</wf:Form>
```

Fig. 5. Polling System example XAML output

5 Conclusions and Directions for Future Research

In this paper we have presented an approach for translating UIs defined in UsiXML to XAML documents with XSL transformations in our preliminary evaluation we have had satisfactory results. The result is a practical solution to the development of platform independent RIA applications as was shown in our study case, the process of reaching final implementation levels needs complementary information not available in more abstract levels while the reconstruction of CUI models is simpler because the process of abstraction is a simplifying task. Right now we are compiling a repository of all the UI gadgets (components) defined in XAML for complete the XSLT translation sheet in a java implemented prototype called RIAXML but in this early stage of

development, some interesting features are still non included e.g., how to deal with alternatives in the widgets selection and define the most suitable.

While we concretize the template model of our application more details have to be included. For instance, width and length of each element, colors, position within the containers, among others. So, how to define this set of attributes? It's a question that requires the aid of ergonomic criteria to be answered. Furthermore, the presented solution is targeted to XAML in future versions, the final implementation code should be a free choice give it to developers.

Acknowledgments. We would like to thank the reviewers for their pertinent comments, the University of Zacatecas (UAZ) and the BHCI lab members (especially to Jose, Juan and Adrian) at the UCL-IAG department for their support in our visit.

References

[1] Miller, J., and Mukerji, J., *MDA Guide Version 1.0.1*, 2003, Object Management Group, Inc.

[2] Bouillon, L., Limbourg, Q., Vanderdonckt, J., Michotte, B., *Reverse Engineering of Web Pages based on Derivations and Transformations*, Proc. of 3rd Latin American Web Congress LA-Web'2005 (Buenos Aires, October 31-November 2, 2005), IEEE Computer Society Press, Los Alamitos, 2005, pp. 3-13.

[3] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt, *A Unifying Reference Framework for Multi-Target User Interfaces*, Interacting with Comp., Vol. 15, No. 3, June 2003, pp. 289-308.

[4] Luyten, K., Coninx, K, and Abrams, M., *Integrating UIML, Task and Dialogs with Layout Patterns for Multi-Device User Interface Design*. The 11th International Conference on Human-Computer Interaction, Las Vegas, Nevada, USA, July 22-27, 2005.

[5] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and Lopez, V. *UsiXML: a Language Supporting Multi-Path Development of User Interfaces*, Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004 (Hamburg, July 11-13, 2004). Lecture Notes in Computer Science, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 207-228.

[6] O'Rourke, C., *A Look at Rich Internet Applications*. Oracle Magazine. July - August 2004.

[7] Berglund, A., Boag, S., Chamberlin, D., Fernández, M. F., Kay, M., Robie J. and J. Siméon, *XML Path Language (XPath) 2.0*. http://www.w3.org/TR/2005/WD-xpath20-20050404/#id-references, (W3C, March 15th, 2006).

[8] XML User Interface Language (XUL) 1.0, http://www.mozilla.org/projects/xul/xul.html (Mozilla Foundation, March 20th , 2006).

[9] Openlaszlo. http://www.openlaszlo.org/ (Laszlo Systems, Inc., March 20th , 2006).

[10] FLEX. http://www.macromedia.com/software/flex/ (Adobe Systems Incorporated, March 20th ,2006).

[11] XAML. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnintlong/ html/ longhornch01.asp (Microsoft, March 20th, 2006).

[12] WinFX SDK. http://msdn.microsoft.com/winfx/ (Microsoft, March 10th, 2006).

[13] Gerber, A., Lawley, M., Raymond, K., Steel, J., and Wood, A., "Transformation: The Missing Link of MDA".

[14] http://www.oxygenxml.com/ (SyncRO Soft Ltd, March 10th, 2006).