XSFORM: A SCHEMA-DRIVEN FORM-BASED XML INFORMATION PROCESSOR

Shang-Hsien Hsieh and Hsien-Tang Lin
Department of Civil Engineering, National Taiwan University, Taipei, Taiwan 10617, R.O.C.
shhsieh@ce.ntu.edu.tw; nicep@caece.net

Abstract: A form-based XML information processor, named XsForm, has been developed in this work. XsForm takes an XML schema as input and generates the corresponding Windows forms automatically to facilitate gathering, editing, sharing, and management of XML information. The information processed in XsForm is then saved into a well-formed and valid XML file for further e-business applications. This paper discusses the development of XsForm and its application to promote e-business in the construction industry.

Keywords: XML, Schema, UIML, Form-based information processing, e-business.

1 INTRODUCTION

To promote information sharing and exchange on Internet among collaborative parties in the e-business process, many industries, including the construction industry, have made considerable efforts on information standardization using the XML (eXtensible Markup Language) technologies [1]. The XML Schema [2] technology, in particular, has been extensively employed to standardize shared vocabularies as well as the structure, content and semantics of commonly used documents for a particular industry or area of application. For example, ebXML [3] has been proposed to enable enterprises of any size, in any location, to conduct business using the Internet; aecXML [4] is being developed to represent information in the Architecture, Engineering, and Construction (AEC) industry.

In addition to the information standards (in terms of standard schemas), an information processing tool is needed for people to gather, digitalize, edit, and save the data for a given XML schema. The most common approach used by this kind of tools to ease the data processing tasks is the use of a form-based Graphical User Interface (GUI). Although the GUI and its associated functions of this kind of tools are similar, people often spend repetitive efforts on developing new tools or modifying/extending old tools whenever they need to deal with XML documents (files) of different schemas.

To help building the XML information processing tools, several related research and commercial efforts have been made. For example, XQForms [5] is a Web-based GUI builder to generate a query form and reports for XML data. GuiGen [6] provides a comprehensive set of tools for creating form-based GUI applications on Windows. Microsoft InfoPath [7] is a commercial product that provides familiar Microsoft Office System environment for the users to create forms solutions, using existing customer-defined schemas, for gathering and editing XML information in the WYSIWYG (What You See Is What You Get) way. In addition, a W3C specification called XForms [8] has been proposed that combines XML and forms to offer a better alternative to HTML-based forms. Several XForms engines have been developed to support the applications of XForms.

Although the aforementioned efforts have provided useful tools to ease the task of building form-based XML information processing applications, fair amount of repetitive development efforts on form-based GUI is still not avoidable when the user needs to deal with XML documents of different schemas. To address this problem, this research has developed a form-based XML information processing application, called XsForm, that is driven by the user-input XML schema to automatically generate appropriate Windows forms and associated functions. This paper discusses the development of XsForm and its application to promote e-business in the construction industry.

2. SYSTEM ANALYSIS AND DESIGN

The major requirements for development of XsForm are the following:

- The system should build an appropriate form-based GUI automatically based on the user-input XML schema to facilitate the gathering, editing, and management of XML information associated with the schema. In addition, the system should allow input of a new schema at any time and rebuild its form-based GUI accordingly and automatically.

- The system should provide some flexibility for the user to adjust the appearance of GUI.
- The system should provide functions for saving the information into an XML file that is well-formed and valid with respect to the user-input XML schema as well as for searching and querying XML information in the XML files managed by the system.

According to the requirements discussed above, the architecture of XsForm is designed as shown in Fig. 1. The arrows in Fig. 1 show the data flow between the user, the XsForm functional components (in the business logic tier and UI tier), and the XsForm functional components (in the data tier). XsForm takes an XML schema as input and uses the Schema2UIML component to parse and transform the XML schema into the corresponding UIML file. UIML (User Interface Markup language) [9], an XML language for defining user interfaces, is employed in this work to describe and record detailed information about the required GUI. The recorded information is then used by the UIML2GUI component to create Windows forms and associated functions. The above process from the input of XML schema to the creation of corresponding Windows forms is performed automatically by XsForm.

The GUI Arrangement component in Fig. 1 provides some options the user to customize the automatically generated form-based GUI. The preferences specified by the user for the options are stored in the Configure file, which is used by the Schema2UIML component to create the UIML file for customized GUI appearance. In addition, the Search GUI component supports full-text search for all XML files in the user-specified file directory and its sub-directories. It also provides customizable advanced search functions for the user to query XML information based on selected tag names in the user-input XML schema.

XsForm can also take a valid XML file as input. In this case, the system finds and loads the corresponding XML schema automatically. The automatically generated Windows forms are then filled with the data of the XML file. XsForm produces XML files as output. All of the XML files produced are not only well-formed but also valid with respect to the user-input schema.

3. SYSTEM IMPLEMENTATION

This work uses the Microsoft VisualBasic.NET with .NET Framework to implement XsForm. In this section, discussions are focused on the implementation of the two most important components in XsForm: Schema2UIML and UIML2Schema.

3.1 Schema2UIML

Schema2UIML is mainly a parser that transforms an XML Schema into a corresponding UIML file. The XML library, called System.xml, provided by the Microsoft .NET Framework is used to implement Schema2UIML.

Schema2UIML should be able to deal with most of the XML Schema syntax (including the nested structure), and map different types of elements in the schema into appropriate corresponding Windows form widgets. Because of the structure of most XML Schemas has no more than three layers (the root element not included) as illustrated in Fig. 2, the following rules can be used to automate the mapping between the XML Schema and Windows form widgets (see also Table 1 and Fig. 2):

a. The root node in the XML schema usually has the name or title of the schema. This research converts the root node into a base Windows form and takes the value of root element as the title of the window.

b. The child nodes of the root element usually represent classification groups, each of which has some sub-elements with similar properties. Therefore, these similar elements are grouped and presented together in the same window by Tab Control or Tab Pages.

c. The elements in the second layer of the schema structure tree contain detailed information of a classified group. These elements have various kinds of data types, such as string, integer, Boolean, and date. They also can be transformed into proper Windows form controls, such as textbox, label, combo-box, and month calendar (see Fig. 3).

d. If the third layer exists, those elements may contain more detailed information than those of the elements in the second layer. The mapping rules are the same as those in the second layer. If there are more than one element belonging to the same parent-element (for example, a person may have more than one phone number), the Group-Box widget is used to enclose those elements.

An example transformation by Schema2UIML is shown in Fig. 4. Figure 4(a) shows the fragment of an example XML Schema. The transformed UIML contents are shown in Fig. 4(b).

3.2 UIML2GUI

The UIML file produced by Schema2UIML has already recorded the structure of user interface and information about what components should be used to represent the elements of different types in the schema. It is then the responsibility of UIML2GUI to construct the actual Windows form widgets based on the description in the UIML.

UIML2GUI is implemented as an MDI (Multiple Document Interface) container to support review and editing of multiple XML files in the same window as well as to facilitate file management and comparison.

Figure 5 shows an example user interface generated by UIML2GUI. In this example, there are 19 pages of TabCobtrols in total on the top of the Windows form. The Windows form controls employed on the selected tab page include the Label, TextBox, and ComboBox, etc. In addition, a grid-style interface is automatically added by the system to assemble data grids at the bottom of each tab page with an "update" button to facilitate the management of data.

4. SYSTEM APPLICATION

An example is provided here to demonstrate how XsForm can be used to promote e-business in the construction industry.

Assume that a set of paper-based forms required in the construction process needs to be converted into a set of electronic forms in order to perform e-business tasks. Also, there is a need for a tool to collect and digitalize data for the set of forms and store the data in the desired XML format. As shown in Fig. 6, the application of XsForm to help the situation in an effective and efficient way is described in the following steps:

1. Define an XML schema to describe the structures, contents, and semantics of the set of forms. Although this step requires some fair amount of efforts, the resulted XML schema is the key document for sharing and exchange of the information in the set of forms with collaborative parties in the e-business process.
2. Execute XsForm and read the XML schema into XsForm.
3. XsForm automatically and quickly generates appropriate Windows forms for the user to digitalize and edit data in the forms. It also allows the user to save the data in the XML files that is well-formed and valid with respect to the given XML schema.
4. The XML files and the XML schema can then be used in the e-business process for information sharing and exchange.

It can be seen that XsForm significantly eases the task of collecting and editing electronic information that is usually the first step toward information sharing and exchange in the e-business.

5. CONCLUSIONS

A schema-driven form-based XML information processor, called XsForm, has been developed in this work to facilitate gathering, editing, sharing, and management of XML information. XsForm generates Windows forms with associated functions automatically according the user-input XML schema. It helps to save repetitive efforts on development of similar but independent form-based applications, each for processing information in XML documents associated with a particular schema. XsForm has also been shown to be an effective and efficient tool to ease the task of collecting and editing electronic information and to promote information sharing and exchange in the construction industry.

REFERENCES

[1] Ray, E. T., *Learning XML*, Second Edition, O'Reilly & Associates, 2003.

[2] Vlist, E. V. D., *XML Schema*, O'Reilly & Associates, 2002.

[3] Gibb, B., and Damodaran, S., *ebXML: Concepts and Application*, Wiley, 2002.

[4] Segarra, S., "aecXML Domain Summary," aecXML Technical Committee, Industry Alliance for Interoperability (http://www.iai-na.org/aecxml/), 2001.

[5] Petropoulos, M., Papakonstantinou, Y., and Vassalos, V., "Building XML Query Forms and Reports with XQForms," *Computer Networks*, Vol. 39, pp. 541–558, 2002.

[6] Reinefeld, A., Stüben, H., Schintke, F., and Din, G., "GuiGen: A Toolset for Creating Customized Interfaces for Grid User Communities," *Future Generation Computer Systems*, Vol. 18, pp. 1075–1084, 2002.

[7] Hoffman, M., "Architecture of Microsoft Office InfoPath 2003," Microsoft Corporation, http://msdn.microsoft.com/library/en-us/odc_ip2003_ta/html/odc_inArch.asp, 2003.

[8] Dubinko, M., *XForms Essentials*, O'Reilly & Associates, 2003.

[9] Abrams, M. and Helms, J. (Editors), "UIML 3.0 Language Specification," Harmonia Inc., USA, 2002.

Table 1 The mapping rules employed by Schema2UIML

XML Schema	UIML
Root node	Windows Form
First layer	TabPages
【String】、【Integer】in the second and third layers	Label＋TextBox
【Boolean】	ComboBox
【Date】	Month Calendar
Third layer element	Group Box

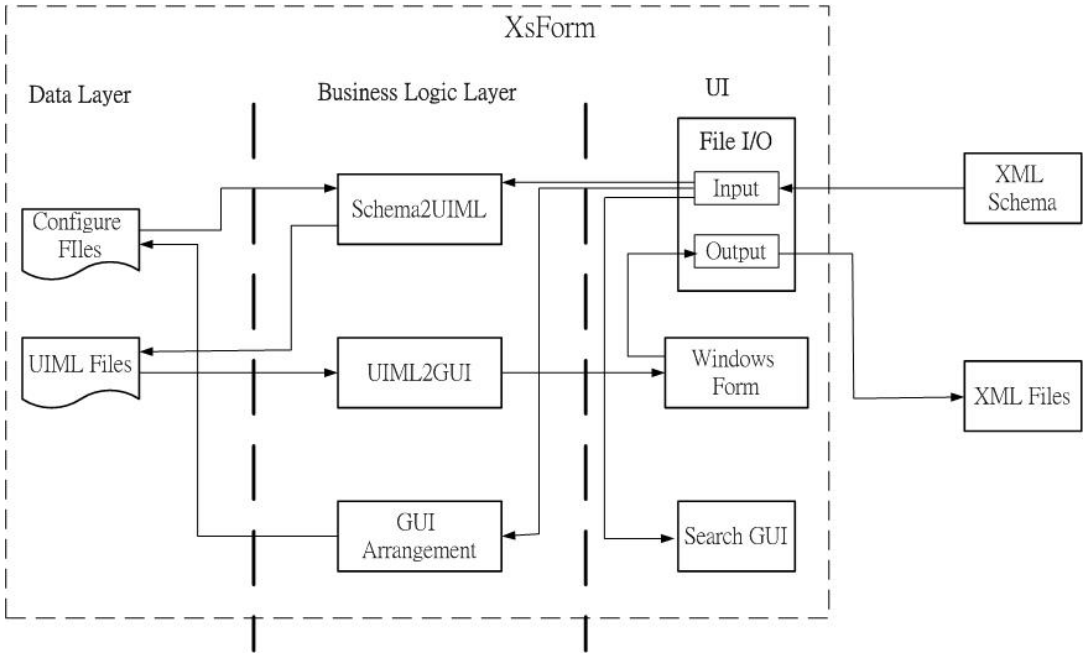

Figure 1 XsForm's Architecture.

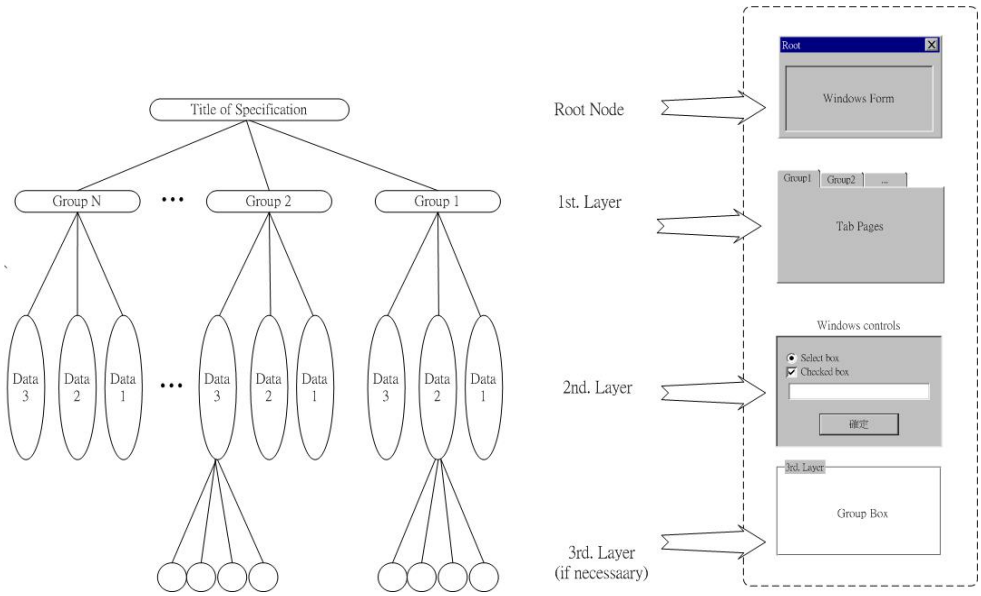

Figure 2 Transformation from XML Schema to UIML.

Figure 3 ComboBox and MonthCalendar GUI

```
<xs:complexType name="Owner">
- <xs:sequence>
    <xs:element name="ID" type="xs:integer" minOccurs="0" />
    <xs:element name="Name" type="xs:string" minOccurs="0" />
    <xs:element name="Agent" type="xs:string" minOccurs="0" />
    <xs:element name="IdentificationNo" type="xs:string" minOccurs="0" />
    <xs:element name="BirthDate" type="xs:date" minOccurs="0" />
    <xs:element name="Phone" type="xs:string" minOccurs="0" />
    <xs:element name="ContactAddress" type="xs:string" minOccurs="0" />
    <xs:element name="Address" type="xs:string" minOccurs="0" />
    <xs:element name="Fax" type="xs:string" minOccurs="0" />
    <xs:element name="Email" minOccurs="0" />
    <xs:element name="FloorDetail" type="FloorDetail" minOccurs="0" maxOc
  </xs:sequence>
  <xs:attribute name="LFT" type="xs:long" />
  <xs:attribute name="Parent" type="xs:integer" />
  <xs:attribute name="Status" type="xs:integer" />
</xs:complexType>
```

(a) fragment of an example schema

```
<part class="TabPages" id="Owner">
  <part class="Label" id="Owner_ID_label" />
  <part class="TextBox" id="Owner_ID_value" />
  <part class="Label" id="Owner_Name_label" />
  <part class="TextBox" id="Owner_Name_value" />
  <part class="Label" id="Owner_Agent_label" />
  <part class="TextBox" id="Owner_Agent_value" />
  <part class="Label" id="Owner_IdentificationNo_label" />
  <part class="TextBox" id="Owner_IdentificationNo_value" />
  <part class="Label" id="Owner_BirthDate_label" />
  <part class="MonthCalendar" id="Owner_BirthDate_value" />
  <part class="Label" id="Owner_Phone_label" />
  <part class="TextBox" id="Owner_Phone_value" />
  <part class="Label" id="Owner_ContactAddress_label" />
  <part class="TextBox" id="Owner_ContactAddress_value" />
  <part class="Label" id="Owner_Address_label" />
  <part class="TextBox" id="Owner_Address_value" />
  <part class="Label" id="Owner_Fax_label" />
  <part class="TextBox" id="Owner_Fax_value" />
  <part class="Label" id="Owner_Email_label" />
  <part class="TextBox" id="Owner_Email_value" />
  <part class="GroupBox" id="FloorDetail">
</part>
```

(b) the corresponding UIML

Figure 4 a Schema2UIML example.

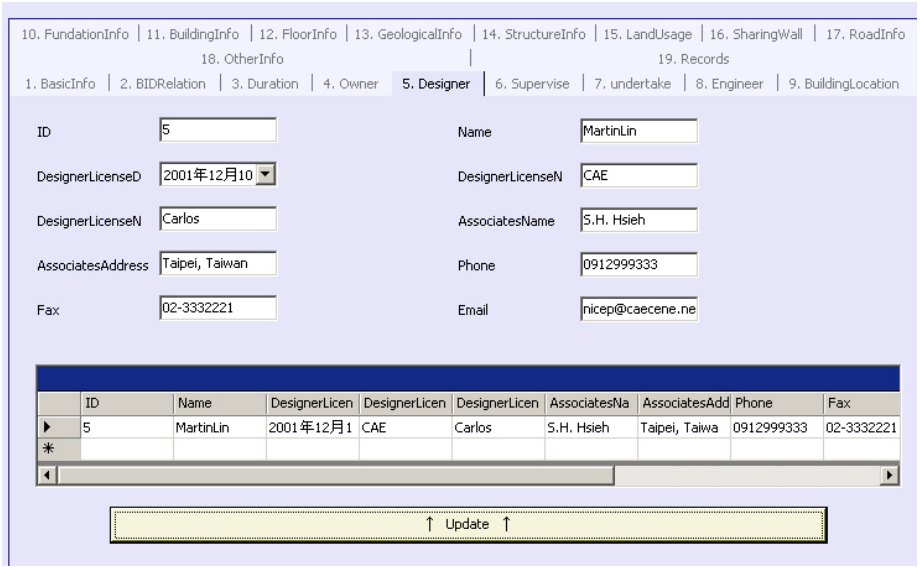

Figure 5 An example user interface generated by UIML2GUI

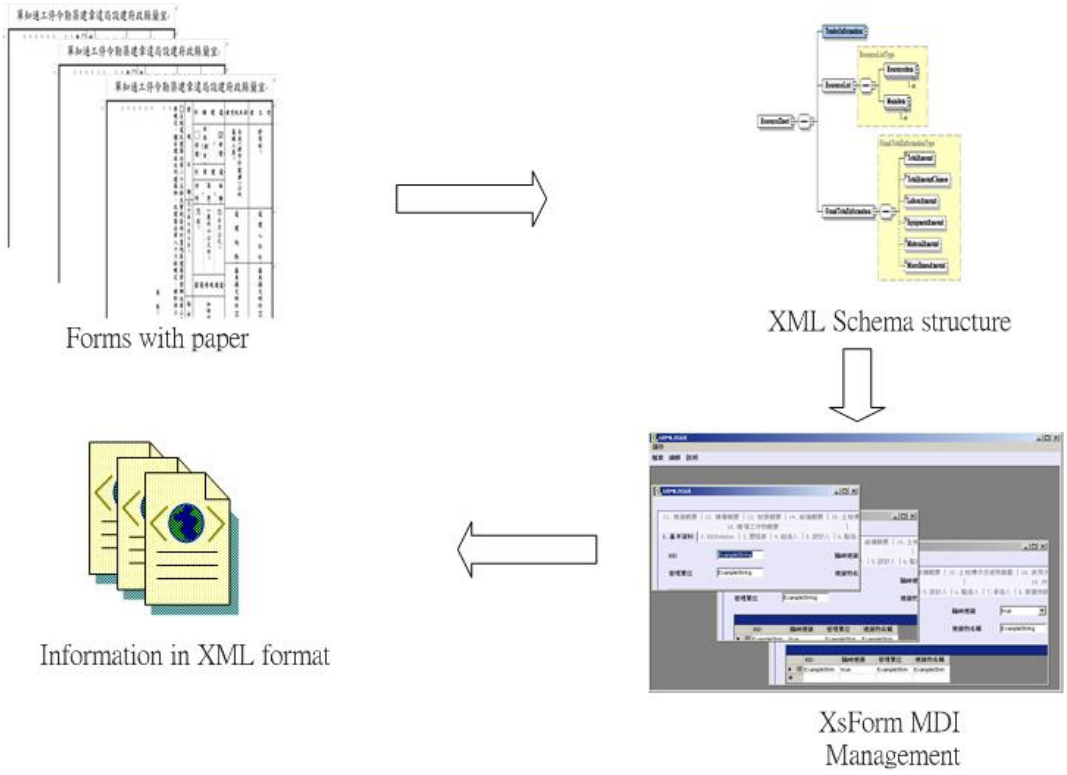

Figure 6 Application of XsForm