World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:10, 2008

User Interface Oriented Application Development (UIOAD)

Mahmood Doroodchi, Babak K. Farahani, and Mahmoud Moravej

***Abstract*—**A fast and efficient model of application development called user interface oriented application development (UIOAD) is proposed. This approach introduces a convenient way for users to develop a platform independent client-server application.

***Keywords*—**Software Development, XML, XForms, XUL, eForm, User Interface.

I. Introduction

TRADITIONAL application development involves the design of data model/ERD diagram, develop the database, implement the business layer, and finally the presentation layer. Some advanced methodologies like RAD, let developers to develop their applications faster by facilitating advanced and special tools. Another paradigm shift in application development seems happening based on developing smarter components [1,5,6] at different layers of software. Consequently, we can expect faster and easier application design/development. This process focuses on the higher level of development or the user interface design. To accomplish the task, we need a set of tools to support this methodology. In other words, smart components would develop the back end of the applications from the GUI, by setting all necessary constraints, validation, business rules and other components.

We call the proposed method "User Interface Oriented Application Development, UIOAD". Previous related works based on similar concepts have been brain-stormed and user interface driven system design [2] has been proposed. But in this approach, since the user interface (UI) forms are essential in capturing data requirements, UI prototype can be used by the developer as a source for developing the classes and objects. The existence of a field on a dialog, a web page, or a report means that the data must either be an attribute of an object, the result of some operation on an object or series of objects, or be calculated from some other object(s) attributes. When the same data exists within different objects on the same user interface, it means that those objects are related to each other and results into an association between classes in the class diagram. Initial repetitions for such associations might be detected by the occurrences of the related objects. [3]

Mahmood Doroodchi is with Cardinal Stritch University, Milwaukee, WI 53217, USA.

Babak K. Farahani, and Mahmoud Moravej are with South Information Technology Co. (www.southit.com), Iran.

Another recent attempt to create a UI centric development is XUL [4]. XUL is designed specifically for building portable user interfaces.

In this paper we are proposing UIOAD concept with the supporting software architecture to create a platform - independent information system from the GUI. In addition, a sample commercial application based on this concept will be introduced.

II. New System Design Process

In this methodology, we start the design of a new system by first designing the user interfaces or forms of the desired application. To accomplish this, use cases are going to be determined first followed by designing the interfaces based on the design of the user interfaces as shown in Fig. 1.

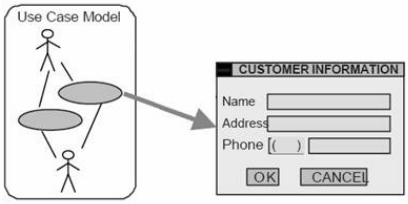

Fig. 1 The design process starts by designing the use cases and the interfaces based on the use cases

Once the use cases and user interfaces are designed, the development environment will build the supporting structure for the new application including classes, objects, and eventually the database and its corresponding objects, transparently. To accomplish that, this methodology includes three main phases: "User Interface Design", "Parsing Mechanism", and "Database Generation". Therefore, we can assume that a system shown in figure 1 can implement this methodology. In other words, such system requires the following building blocks: "User Interface (UI) Designer", "Parser Engine", and "Database Factory" as shown in figure 2.

Each of the shown phases is explained in more details in the following subsections.

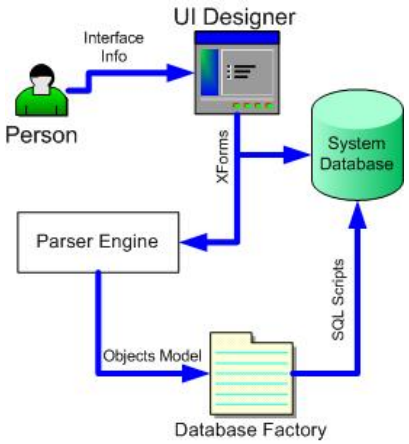

Fig. 2 The entire process of designing a new system

A. User Interface (UI) Design Phase

This phase includes the following parts:

- **Determination of Form role in system (Systematic viewpoint)**
  In this part, the flow of information from each interface will be determined. This is done by determining each form's role in the system and the type of users or actors that use each interface. Issues such as access control and security of the users are also addressed here. Knowing the role of the form in the system means that we try to find out how this piece of the puzzle affects the entire picture. Use cases interoperations and interactions are to be determined and the information flow for each case is specified.
- **Form Detail Determination**
  This stage includes declaring the form details. This high level definition will help us not only to decrease our dependency to different platforms but also to make it possible for high level users to design a form, so that they could describe their forms in an abstract form. There are some well known innovated tools/standards that are suitable for this methodology such as XForms standard. It will bring a distinct part in conjugation with XML standards to design a form. Another example is XUL. Generally our output can be one of the HTML, XForms, XUL, MS Word, or Pdf formats.
- **Interface Functional Definition**
  In this totally conceptual stage, an interface is defined by defining its functionality. This is done by answering a series of questions as shown below:

  - What kind or type of information does the form get?
  - What kind of necessary constraints does a particular field in the form have?
  - How do the form objects relate to the information in other forms? (For example contents/items in the combo boxes, option lists, …)

In this part another important aspect of XForms is discovered. This, in conjugations with other XML standards, provides capabilities to declare the form model absolutely independent of the logic behind the interface.

B. Analysis and Object Production Phase

The output of the previous phase is an XForms document, which includes the design and logic of the form. All the designed items are coming from a standard template. The XForm document maintains the design and the logic of the form as a standard relation between Design and Development phase.

This is the main phase in our proposed method. It receives the structured input in the special format of XForm, and transforms it to a well-defined format of the programming languages. In fact this part will parse text into objects as declared. This part will generate the data model as a data structure, considering the input fields of the form. Also, proper communication means to other classes and objects are considered in the class definition. The methods are defined in a very abstract and general form since the classes are made dynamically.

C. Database Generations

After generating the classes and the relational model, the database should be developed. Database Factory is responsible for transformation of generated structures to the equivalent model in a database as shown in Fig. 3.

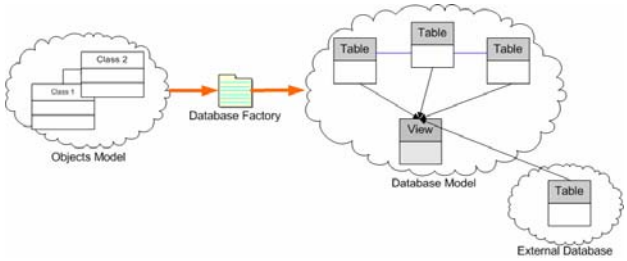

Fig. 3 Extracting the ER model from designed forms

Transforming of the OO Model (Which has been done by parser engine) to an ER model is a challenging process and will be done by database factory. In this phase, inputs and outputs are the OO structures and relational structures, respectively.

III. Methodology in Action

Once the system is built, the system information will be stored in the system database. The first step to use the system is to retrieve the form information and rebuild it for the user. The next step is to get the data from the user and store the information in the database.

A. Form Displaying/Rendering Process

This process includes presenting the form to the user independent from the platform depending on the environment. For the form inside a browser will be displayed using HTML.

"Form Displaying Process" can be summarized in Fig. 4 that will be explained later.

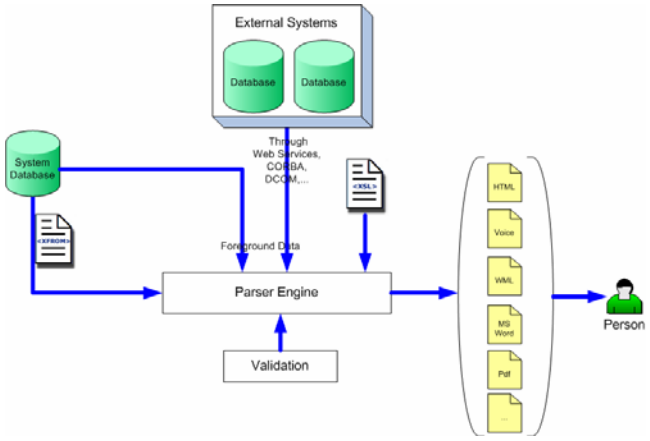

Fig. 4 Forms displaying/rendering process

One of the important outputs of this process is an XForm document that has the exact declaration of the original use cases in the design phase. This declaration is abstract and platform independent. As we saw in design phase, parser engine is responsible for converting declarative model of designed form to OO model. The other responsibility of parser engine is to generate the actual user interface from the stored information and according to end user's platform.

The parser receives form's declarative definition in XForms format and then regenerates the user interface for the target platform from the logic and the user interface design information. For example for regular web browsers it generates HTML codes while for PDAs it may generate WML codes or a windows form for PC desktop applications. To perform this task, parser uses the following parts of the system or information:

XSL files and XSLT files: These files have the main role in generating the interface and the look of the form for different platforms.

Database: The parser may need to establish a connection to the database to get some information about the sub forms or sibling forms and use them to generate the code.

Validation: In addition to the user interface code, some auxiliary-codes maybe needed to control/validate the input data like by injecting javascript control codes into the HTML forms.

Web Services: If a form requires to be connected to another distributed system, then web services is the access method to such system.

In this way, the form will be displayed and is ready for the next stage.

B. Processing and Storing Data

The user fills the displayed form and returns it to the system. Fig. 5 shows the steps clearly (it assumed the form is an HTML one):

After filling the form by the user, the data will be delivered to parser engine box. One of the other responsibilities of this box is converting user's data to the proper format that would be tangible by the system (it has different responsibilities in "designing form" and "displaying form" phases). It should recognize the relations between sent data and form's fields by referring to the form's definition document (in XForms format). This is the first step in importing user's data to system as a structured one. The next step, which is so important, is data validation. In "displaying form" process, the validation was done in partial mode but the main activity would be done after receiving user's data. In this step, user's data should be validated against the form's logic. In other words the declarative statements that would be supplied in form's logic definition would be checked against the user's data to verify if they are consistent with them or not. But as we mentioned earlier, it may be impossible to validate the user's data at the same place where we display the form to the user. Then it is required to connect to the other part of same system (or even other systems) to verify correctness of user's data. So as you see in the Fig. 4, the validation box may connect to database or use some web services (to connect to other systems in distributed solutions) to see if the user's data are consistent with other data or not. Note that we can connect to other systems with different solutions like using RPC (Remote Procedure Call), CORBA (Common Object Request Broker Architecture). DCOM (Distributed COM) and so on that are not mentioned in this diagram.

Here we have a consistency problem between distributed databases that should be resolved by the DBAs (Database Administrators). Of course this issue is out of this article's scope and depends on system designation.

After user's data is verified and relations between data and form's components are found, the related objects would be created from the classifiers that were made in "designing form" phase. In other words, the user's data would be stored in some objects (in OO concepts). Therefore the user's data are capable to circulate through the system easily and being processed wherever is needed because its format is recognizable for the program generator system. Briefly, the objects are the real instances of form's data in the format of the classes that present the form's schema. These objects are outputs of parser engine in this phase that could be used as input data for other system parts later.

Finally these generated objects should be stored in database. This process would be done by "Data Access Methods" part. This part receives the user's data in object format and then stores them in database as output result. This part is responsible for mapping these received objects to such database objects that were created in "design form" phase and store received objects data into these database objects. Note that the reason for this mapping in current phase is because of input objects of this box are OO objects whereas the equivalent objects in database are Database Objects (like tables, views, stored procedures and so on).The received objects would be mapped to related ones in database system as a result. Moreover, "Data Access Methods" routine should store related user's data information to make it available for future access and also for specifying the state of user's data in system. This information usually consists of one unique identification number, date of filling form, related actors and so on.

C. Representing the Filled Forms

This stage consists of representation of a designed form with data samples. In other words, it is the final stage of a designed form including data.

As mentioned before we can divide this stage into two major steps:

- Form Representation
- Injecting form's data to the rendered form

At the first step, design part of the form in XForms document (declarative, platform dependent) is converted into as implemented model in target platform. Obviously in this conversion would be done by XSLT. Also for preparing required form data which are dependent on other parts of the system such as other forms or even other software systems, we use web services. The second part includes the following steps: based on XForms, related entities in database are recognized. Data Access Component does this task, which has been used in data store step, too. This component gets the XForms and complementary information and retrieves the filled forms data from the database.

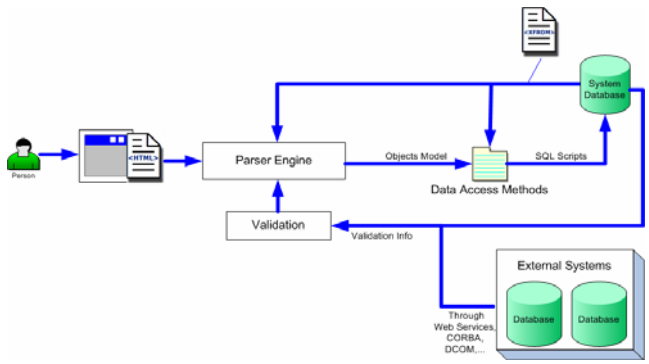

Fig. 5 Processing and Storing completed forms

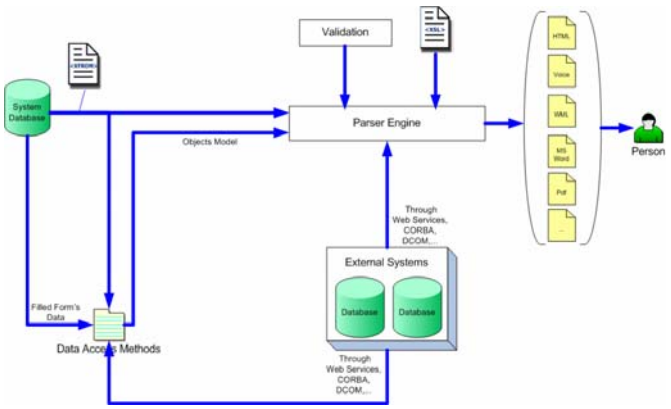

Fig. 6 Representing the filled forms

Major parts of the retrieved data in previous step are unprocessed. So it is necessary to retrieve more data from database or web services to prepare the rendered form's data.

The DAC, creates the OO structure for rendered data by using XForms structure. After this stage the form's structure would be available in system as a set of controls/components dependent on platform.

The rendered controls would be transfer to the parser engine. This engine:

- Renders the form for platform.
- Maps the prepared data to controls.

The completed form would be presented to the user.

IV. An Implementation of UIOAD

South Information Technology Co. (Sitco) is the largest software vendor in south of Iran. Sitco eForm has been developed to be a tool for developing new applications based on UIOAD concept. The application has been developed as an n-tier service oriented software based on Microsoft .NET Framework. It uses Microsoft SQL Server as its database engine for storing all application related information, however it woks with any other RDMBS system such as Oracle. Sitco eForms is a 100% web based application and covers almost all UIOAD concepts. As a best practice Sitco eForm uses Windows Servers Active Directory Service Services as its user and workgroup management engine to provide a low TCO and high ROI solution and reduced administration cost across the enterprise.

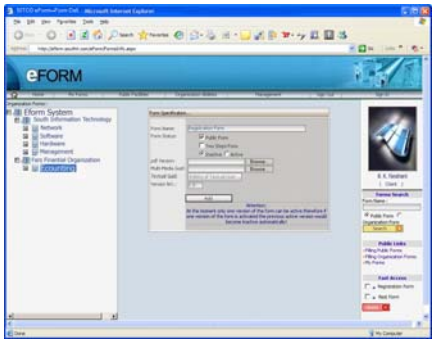

Fig. 7 Sitco eForm Form definition page

Its form designer is developed with Macromedia Flash Designer and generates designed forms as an extended XForms schema. Most schema extensions were implemented for supporting right-to-left functionality. The designer supports most of common validations and constraints.

Fig. 8 Sitco eForm Form Designer

Sitco eForm supports forms relationship concept, which gives the eForm developers the ability of defining general forms as global forms and use their information in other forms, however it supports sub form usage as an advanced

feature. Global forms information would be render as common list based controls like radio button lists, check box lists, list box, combo box, data grid and others.

The application generates HTML web pages as .NET ASPX pages with its XForms server side ASPX engine. Since the application generates ASPX pages on the fly, it utilizes all .NET CLR features such as validators, server controls, and others. The product supports form's element constraint definition, onBeforeSave and onAfterSave event as a key feature to develop real world applications based on UIOAD.

Fig. 9 Sitco eForm General Forms Information relationship and Forms

Fig. 10 Sitco eForm rendered form sample

V. Conclusion

A new methodology is proposed in this paper to develop a system from the user interface without being worried about developing the supporting databases.

References

[1] L. Robert Varney and D. Stott Parker, "Generative Programming, Interface-Oriented Programming and Source Transformation Systems" .
[2] D. Batory and B. J. Geraci. Component validation and subjectivity in GenVoca generators. IEEE Transactions on Software Engineering, pages 67 - 82, 1997.
[3] Stevan Mrdalj, User Interface Driven System Design, Eastern Michigan University.
[4] http://www.xulplanet.com/tutorials/xultu/intro.html
[5] M. Fowler, (1997). Analysis Patterns: Reusable Object models, Addison-Wesley.
[6] How to develop an application with Sitco E-Forms, Sitco Internal Document.
[7] Sitco eForm Concepts and Architecture, Sitco Internal Document.