HDE-UITool: A Rapid Interactive Interface Prototype Generation Tool for Aircraft Design Software

Haiyang Bai, Jing Li, Peiyun Zhou, Yi Zhuang
College of Computer Science and Technology
Nanjing University of Aeronautics & Astronautics
Nanjing, China
e-mail: bhyself@163.com

***Abstract*—In the requirement analysis phase of software engineering, an interactive interface prototype system can intuitively reflect the results of the current design and bring benefits for further requirements communication. Aircraft design software has the features of high number of input/output parameters and developing such a prototype system usually costs much time. According to the goals of rapidly designing and generating such an executable user interface (UI) prototype, based on the data model and XML technology, a rapid UI design and generation tool is represented. In this tool, functions of rapid data binding, interface customization are realized by the form designer and functions of automatic generation, dynamic modification and event handling are achieved by the UI generation engine. At last, an example is given to show that this tool can quickly and flexibly complete data binding, interface design and generation work, and greatly shorten the development time of the prototype system.**

Keywords- Aircraft design software; prototype system; XML; UI design and generation;

I. INTRODUCTION

In the process of software engineering, a prototype system with several main UIs and simple interactions allows the early validation of the system model and help in requirements elicitation[1]. It is also useful to make clear the data arrangement and workflow of the design software in the early stage of software engineering. However, UIs of aircraft design software have the features of high number of design parameters[2] and relatively complex data arrangement structure. Data binding and layout management usually cost much time. In order to simplify the UI design and generation process, reduce the runtime overhead, and keep the interface flexible adjustment, a rapid UI design and generation tool, named HDE-UITool is developed.

This tool simplifies the job of mapping data to controls. The form designer and UI generation engine realize functions of UI customization, automatic generation, dynamic modification and event handling. Interactive process can be costumed to show the logical relationship between the UIs. XML is used as the storage media to store UI information, and XML Schema is defined to describe the format of the XML files.

In this paper, the HDE-UITool is described to support the rapid UI design and generation. The rest of the paper is structured as follows. Section 2 reviews the related work about UI design and generation. Section 3 describes the workflow of UI design and automatic generation. Section 4 presents key technologies and implementation of form designer and UI generation engine, which are the main work of this paper. Section 5 shows the test result of the HDE-UITool and analysis the effect of it. The last section concludes the paper and outlines some directions of future work.

II. RELATED WORK

There are currently numerous and various approaches using different input materials to UI design and automatic generation such as techniques based on design patterns, use-case scenarios and models[3, 4, 5].

Borchers firstly proposed a formal representation of patterns and their relations in the field of HCI[6]. Considering the limitations of model-based user interface design, with a view to fostering reuse in the instantiation and transformation of models, Gaffar introduces patterns as building blocks, which can be first used to construct different models and then instantiated into concrete UI artifacts[7]. Stefan et al use XML-based user interface description languages like UIML and UsiXML for the specification of user interface patterns[8].

António gives an approach[9] for the automatic generation of functional user interfaces prototypes from early system models with minimum effort for model validation and requirements elicitation and validation purposes. The start points are the system's domain and use case models enriched with OCL constraints. Paulo and Norman introduces the notation of the Unified Modeling Language for Interactive Applications (UMLi)[10, 11], that extends UML, to provide greater support for UI design. UI elements elicited in use cases and their scenarios can be used during the design of activities and UI presentations.

Model-based systems attempt to formally describe the tasks, data, and users that an application will have, and then use these formal models to guide the generation of the user interface[12]. Interface model can be roughly divided into the conceptual model and declarative model. Typical conceptual models are Seelleim model[13] (firstly proposed by G.E.Pfaff et al), MVC model[14] and PAC model[15].

Declarative model describes the model from the perspective of software engineering. It has a variety of models to meet the needs of the user interface. The most frequently used models are task, domain, user, dialogue and presentation models[16]. In addition, from a software engineering perspective, there are some other user interface models. For example, the FVI model which is composed of Frame Controller, View Model and Core Interface, supports object-oriented requirement specification and provides a layered, modularized and iterative Object-Oriented GUI design model. EIP[17], which consists of three parts, EOM (Extended Object Model) as E, IM (Interaction Model) as I, and the Presentation Model as P, can supply conceptual support and lay implementation foundation for UI engineering development.

XML is an ideal storage interface language as it is flexible and general. Based on this technology, Kollar propose graphics user interface markup language(GUIML) to describe user interfaces using DOM for accessing UI definition. Some automatic generation designs are based on the XML storage technology, such as the method in paper [18] and in Patent [19].

III. SYSTEM STRUCTURE OF HDE-UITOOL

The rapid UI design and generation flow is shown in Figure 1. In the UI design stage, aircraft design database is firstly connected to the form designer. Parameters in the database are read out and the selected items are mapped with controls by the form designer. XML is selected to be the storage media, so that designed UIs can be permanently stored in files, Besides, an XML Schema(XML Schema Definition: XSD) file is especially designed to standardize the format of XML files. At the end of this stage, an XML resource file containing all the properties of controls and the layout style information is generated.

In the UI generation stage, the UI generation engine automatically generates the UI according to the properties and layout information read out by the XML parser. Properties of controls generated in the form can be modified directly, and the modified part will be stored in the XML resource file again.

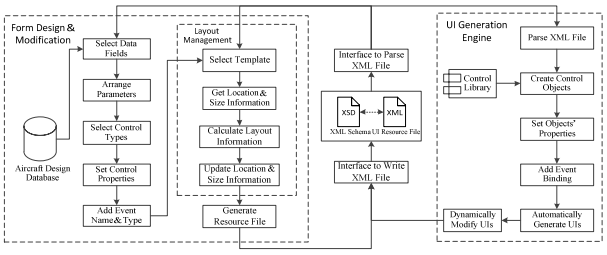

Figure 1. The workflow of the HDE-UITOOL.

IV. KEY TECHNOLOGIES AND IMPLEMENTATION

A. UI Storage Media

(1) XML. The eXtensible Markup Language, XML, has become an increasingly significant part of the IT mainstream, and since its development, XML has been gaining a general acceptance as a standard for exchange information, data representation, and data transmission between distinct applications running on different platforms. XML was created to structure, store, and transport information. To take the advantage of the good scalability, sine descriptive, cross-platform, storage tree structure characteristics of the XML, this paper raises XML-based user interface management model. XML is used to store, describe and manage UIs.

(2) XML Schema. The purpose of an XML Schema is to define the legal building blocks of an XML document. It provides a means for defining the structure, content and semantics of XML documents. Through the XML Schema, a UI conceptual model is constructed. Elements in the Schema are mapped with the controls of the UI. It defines the control types and the properties of them.

Part of the XML Schema structure diagram for this tool is shown in Figure 2. The structure of the schema is organized as a tree. The root node ControlWnd is regarded as a complete UI. It can be divided as several groups, and each of them contains different kinds of controls.

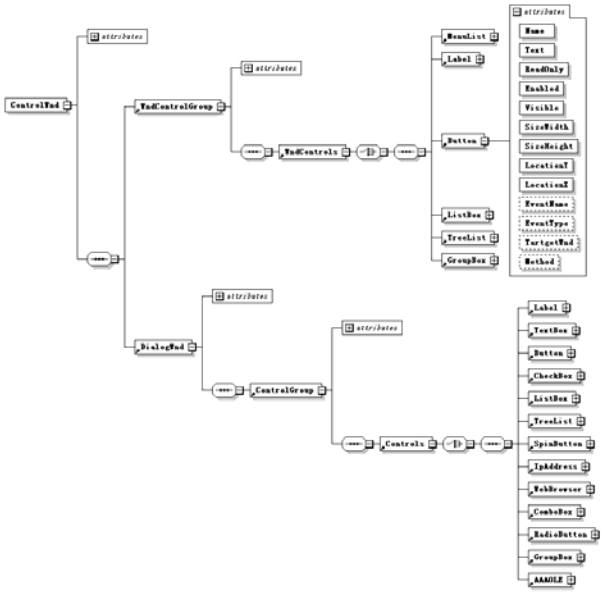

Figure 2. XML Schema structure diagram for HDE-UITool

(3) XML Access Interfaces: XML DOM. In this tool, interfaces are defined by the technology of DOM(Document Object Model, DOM) to write and parse XML files. The DOM is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document. The XML DOM defines the objects and properties of all XML elements, and the methods (interface) to access them. In other words: The XML DOM is a standard for how to get, change, add, or delete XML elements.

Interfaces are designed as an adapter to converse information between UIs and XML files. The designed UI

information's formats are transformed to XML tags which are validated against the XML Schema by the interface. Besides, the parse interface gets XML file's elements and restores them to UI information. Form designer and UI generation engine don't have to know the detailed format of the XML, and it is flexible for the tool to expand or change the UI conceptual model.

B. Form Designer

The form designer is individually developed to design UIs rapidly and it has functions of data binding, arranging controls and layout management. Interface design results are saved in an XML files and they can be read out to be modified and stored again.

(1) Data Binding and Arranging Controls. In order to add batch controls, the form designer offers the function of connecting the database, so that controls' name mapped with strings in tables can be import to the designer easily. By connecting to the airplane design database and arranging the parameters, data model of the UI is constructed. According to the selected data tables and property fields, the program adds the strings to the source item list. The chosen items will be moved to the control list and added to the queue that the program uses to manage control objects.

Except those controls binding with parameters, some other controls such as Button and Label can be manually added. Common controls contain basic properties such as name, size, location, visible, enable, etc. Each of them has proprietary properties and events. All these properties and events set are declared in an XML Schema. In the designer, by selecting control types, detail properties and events setting interfaces matched with the XML Schema are displayed to users.

(2) Layout Management. Layout-manage function offers some frequently-used layout styles such as horizontal and vertical layout template. Based on the group arrangement, control type and number of controls, the layout-manage function computes the size of each control and mass updates the location properties of the listing controls. The preview function shows the main outline of the UI with layout style selected. Actually, it is realized by extracting the size and location information from listing controls to draw graphics onto a dialog. Besides, creating new user-defined styles and the preview function are supported.

C. UI Generation Engine

In order to automatically generate the UIs designed by the form designer, referring to the method in [20], the HDE-UITOOL tool achieves these goals by using a single dialog and implementing a UI generation engine.

(1) UI Automatic Creation. According to the XML Schema, the XML file is parsed by the parsing interface and the detail information is read out. Control objects stored in the XML file are created by the factory method, and they will be put to a container. The container manages the created instances and decides when to destroy them. To implement automatic generation mechanism, the container keeps track of control instances just like a pool. It collects objects of automatically generated controls and attaches to the form view or dialog. Once a control object is created and added to the container, the properties of the control are applied to this instance. At last, the attached view or dialog was created and shows in the window. When the UI is closed, instances in the container will be released.

The classes of the UI generation engine are mainly divided into three parts according to the function relationship. They are control & event classes of new definition, frame & dialog classes and the functional classes for automatic generation method.

The redefined control classes include controls' properties, creation method and event response methods. Frame & dialog classes contain the basic framework of the MFC (Microsoft Foundation Classes). The UIs display in the normal mechanism like frame views or dialogs. The functional part realizes functions of creating control objects and put them to the container.

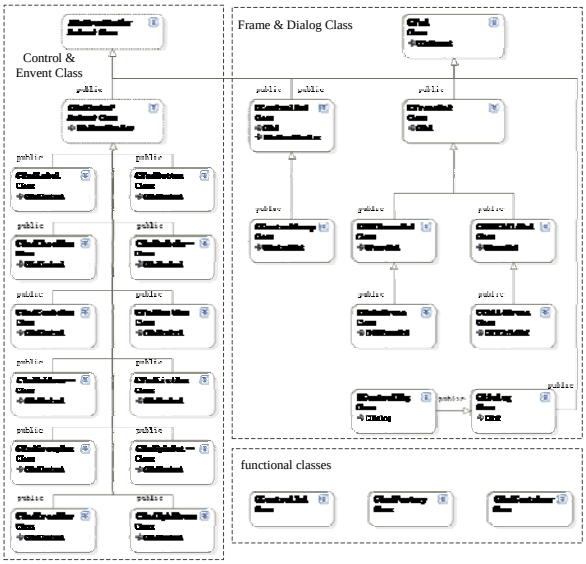

Figure 3. UI generation engine's classes.

(2) Dynamic Modification. In the UI generation stage, for the executable prototype system, property pages make it possible to modify controls' properties dynamically. Different property pages for each control are designed. If one or more properties are modified in the generated UI, the control will be recreated so that the latest effect can be shown in real time. For the program, the instance of the modified control is firstly destroyed, and then created according to the modified property information. Besides, XML items of the old control will be found and replaced by the newly modified information. It avoids returning to the design stage to design a UI and generate an XML resource file again. The modification of the prototype system can be realized dynamically and immediately.

(3)Event Handling. Considering the requirements to act the workflow in the prototype system, some simple event handles are supported in this tool. In the UI design stage, based on the XML Schema, the form designer offers some frequently-used events for each control. However, the method should be declared in the program before, and the event name should be matched with the other one marked in the XML file. The automatically created control objects' messages can be intercepted by overriding the virtual method in the prototype program. By binding events and methods, the executable prototype system may correspond to some external events. Through this way, some basic operates such as switching forms and going to another view can be realized easily. Such a process is shown in the following figure.

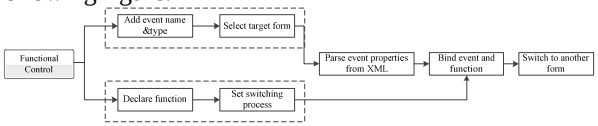

Figure 4. The process of workflow demonstration

V. TEST AND EXAMPLE

The following example shows how a simple prototype's UI is designed and generated rapidly. The form designer connects to the aircraft design database, selects the items needed for the UI and generates an XML resource file. Then, the UI generation engine automatically generates the selecting interface and shows it as a dialog in the window. The finally UI is shown below in Figure 6.

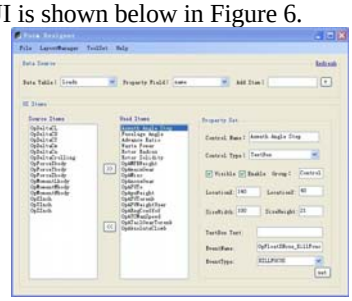

Figure 5. Form designer's main form.

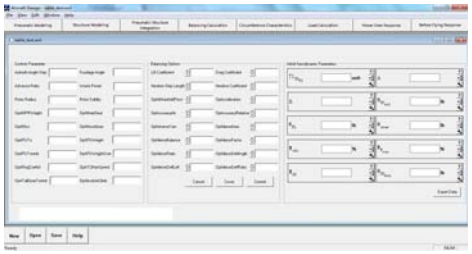

Figure 6. An example of a generated UI prototype system

VI. CONCLUSION AND FUTURE WORK

This paper describes a rapid UI design and generation tool HDE-UITool to build a prototype system of the aircraft design software. We describe the features and implementation methods of different functional unit, which are UI customization, automatic generation, dynamic modification and events handling. A design example proves that this tool can accomplish fast and flexible data binding, interface design and generation work and it greatly reduces the interface prototype system development time.

Future work is needed to enrich the HDE-UITool. Combining the form designer and UI generation engine may reduce the workload of design iterations. At present, the layout management is not so flexible and event handling mechanism is maybe too simple to achieve complex calling functions.

REFERENCES

[1] Agarwal B B, Tayal S P, Gupta M. Software Engineering and testing[M]. Jones & Bartlett Learning, 2010.

[2] Ziemer S, Glas M, Stenz G. A conceptual design tool for multi-disciplinary aircraft design[C]//Aerospace Conference, 2011 IEEE. IEEE, 2011: 1-13.

[3] Julien D, Ziane M, Guessoum Z. GOLIATH: An extensible model-based environment to develop user interfaces[M]//Computer-Aided Design of User Interfaces IV. Springer Netherlands, 2005: 95-106.

[4] Pinheiro da Silva P, Griffiths T, Paton N W. Generating user interface code in a model based user interface development environment[C]//Proceedings of the working conference on Advanced visual interfaces. ACM, 2000: 155-160.

[5] Mahfoudhi A, Abid M, Abed M. Towards a user interface generation approach based on object oriented design and task model[C]//Proceedings of the 4th international workshop on Task models and diagrams. ACM, 2005: 135-142.

[6] Borchers J O. A pattern approach to interaction design[C]//Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques. ACM, 2000: 369-378.

[7] Ahmed S, Ashraf G. Model-based user interface engineering with design patterns[J]. Journal of Systems and Software, 2007, 80(8): 1408-1422.

[8] Wendler S, Ammon D, Kikova T, et al. Development of Graphical User Interfaces based on User Interface Patterns[C]//PATTERNS 2012, The Fourth International Conferences on Pervasive Patterns and Applications. 2012: 57-66.

[9] Rosado da Cruz A M, Pascoal de Faria J. Automatic generation of user interfaces from domain and use case models[C]//Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the. IEEE, 2007: 208-212.

[10] Da Silva P P, Paton N W. UMLi: The unified modeling language for interactive applications[M]//<<UML>> 2000—The Unified Modeling Language. Springer Berlin Heidelberg, 2000: 117-132.

[11] da Silva P P, Paton N W. User interface modeling in UMLi[J]. Software, IEEE, 2003, 20(4): 62-69.

[12] Nichols J, Faulring A. Automatic interface generation and future user interface tools[C]//ACM CHI 2005 Workshop on The Future of User Interface Design Tools. 2005.

[13] G.E.pfaff, P.J.W.ten Hagen, User Interface Management Systems[M]. Springer-Verlag Berlin, 1985.5.

[14] Liu X. Automatic Interface Generation Algorithm Based on the Features of Colorectal Cancer Medical Record[C]//Proceedings of the International Conference on Information Engineering and Applications (IEA) 2012. Springer London, 2013: 179-186.

[15] Lin S S, Shin S S, Hsieh M C, et al. MDA-Based UI Modeling and Transformation of Spoken Dialog Systems[C]//Hybrid Intelligent Systems, 2009. HIS'09. Ninth International Conference on. IEEE, 2009, 1: 47-51.

[16] Schlungbaum E. Model-based user interface software tools-current state of declarative models[J]. 1996.

[17] Jiancheng W, Xudong L, Lei L. A model of user interface design and its code generation[C]//Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on. IEEE, 2007: 128-133.

[18] Feng Wentang, Hu Qiang. XML-based Interface Automatic Generation[J]. Application Research of Computers. 2006, 9: 75-77.

[19] Lin G, Xu X. Method for dynamically generating a user interface from XML-based documents: U.S. Patent 6,941,521[P]. 2005-9-6.

[20] Nschan. Building a Dynamic UI using a CWnd Free Pool [EB/OL], http://www.codeproject.com/Articles/17043/. 2007-1-6/2013-5-12.