Model Driven Development of Complex User Interface

Xudong Lu

School of Computer Science and Technology, Shandong University 73#, Jingshi road, Jinan Shandong Province, China 86-531-88392498

dongxul@sdu.edu.cn

Jiancheng Wan
School of Computer Science and Technology,
Shandong University
73#, Jingshi road, Jinan Shandong Province, China
86-531-88392498

Wanjch@sdu.edu.cn

ABSTRACT

To promote the Model-Based software engineering development of user interfaces, this paper proposes a model driven development approach of complex user interface. The approach captures the process data in user interfaces by using an Extended Object Model. User interfaces are directly, abstractly depicted as objects, components and their cooperative relations in an Interaction Model. Their external visual presentation is customized in the Presentation Model in the name of UI Template. The uniqueness of this research is a direct and leveled abstractive attack on complex user interface composition or description, rather than the elicitation from task models.

Categories and Subject Descriptors

D.2.1[Requirements/Specification]: Methodologies - Model Based User Interface; Representation - User Interface; D.2.2 [Design Tools and Techniques]: Computer-Aided Software Engineering (CASE) - Model Based User Interface Development Environment;

General Terms

Design, Experimentation, Theory

Keywords

Software Engineering, User Interface Design, Code Generation

1. INTRODUCTION

The user interface of an application is often one of the core factors determining its success. The model-based user interface development technology aims to provide an environment where developers can design and implement user interfaces (UIs) in a professional and systematic way, more easily than when using traditional UI development tools [1]. To achieve this aim, many approaches for UI design and model-based user interface development environments (MB-UIDEs) have been proposed. But, in spite of more than 10 year's researches, there are few models and tools being fully developed and powerful enough to be recognized for commercial and industrial acceptance [2]. In the authors' view, the most reason of this situation is that the approaches had been proposed have not enough ability in deal with complexity of user interface.

The complexity of user interface mainly lies in two aspects as following.

First, in user interface of the practical software, there always have more complex UI controls of grid, graph and tree, sharing presentation space that can overlapping many present units so can show different content in different context, and their operational relations.

Second, the contemporary UI construction needs support of different techniques and various types of component and their relationships, such as Java Applet, Java Beans, CORBA and COM/DCOM component, etc..

The complex UI controls and components' introduce enhancing the UI functionality, but they do have alleviated the UI development difficulties in great deal. In this case, lacks in this support of UI modeling will affect the ability in dealing with UI complexities, and will not be accepted in practical applications.

2. RELATED WORKS

The literature contains many papers describing model-based user interface approaches. Reported researches includes Drive[3], MOBI-D[4], Wisdom[5], Teresa[6], Teallach[2], JUST-UI[7], SUPPLE[8], etc. These approaches have respective characteristic as their different applying backgrounds such as web applications, multi-devices applications. But, there still have some commons. For example, the approaches usually have several models describing different aspects of the UI. The kind of models used in different approaches varies, however a useful categorization is presented in [1]: Application Model (AM), Task/Dialog Model (TDM), Abstract Presentation Model (APM), Concrete Presentation Model (CPM).

The UI development process is normally an incremental process [1]. In most approaches, UI development starts from domain objects or concepts likes object, describing the structure and behavior of UI. Next, abstract user interface will build base on domain objects and adding some user interface compose elements, such as presentation objects or query objects etc.. Last, code will generate in some approaches. But, the UIs generated are simple and not be accepted in practical applications because of the UI models not cover the complex elements of user interface mentioned above.

The task model is widely used and most researches adopt one of different forms of task model as the basis of UI modeling with some differences, such as TERESA, Teallach, Trident and Bisignano. But, the task-oriented UI design analysis and composition could not fully reflect users' synthetic and multiple views toward UIs. Nowadays highly usable UIs usually contain more than just the completion of user tasks. Extensive information navigation, cross-reference and context-sensitive help are all necessary facilities in UIs. A form-based UI has supplied an environment for fulfilling tasks.

In conclusion, the declarative models of model-based user interface development approaches have been proposed are still in lack of a sound mechanism for direct describing complex UI structure and composition.

3. ARCHITECTURE

Figure 1 shows the architecture of our approach, which reflects an incremental development process. The process started from the bottom abstract, layered combine the increasing requirements (see the dotted line box), by model transformation get more specific model and as the foundation of next layer. The process emphasized the importance of human in the development of the model, multi-layer transformation support the hierarchical model and hierarchical development, which reflects the actual process of practical software development. In this paper, we only focus on how to direct describe the complex user interface use EIP model, not touch user interface requirements acquisition and final automatic generation as the limited length reason.

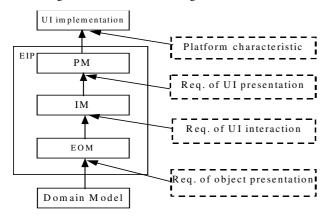


Figure 1. Architecture of Approach

The EIP model consists of three parts, EOM (Extended Object Model) as E, IM (Interaction Model) as I, and the Presentation Model as P.

EOM is the application model of UIs. It specifies the data object and their operational relationships. Considering their influence of objects' very structures and their relations to UI presentation, and the satisfaction of the requirements for describing complex UIs, we extend domain model through adding new attributes and relations for describing data members and their behavioral relationships.

Based on EOM and user's requirements of UI interaction, IM acts as a mechanism for direct description of UI. It includes functions, abstract objects, components, external UIs, and their interoperational relationships, and interactive relations with users.

P is the Presentation Model of UIs. It gives the UI layout according to the IM and user's requirements of UI presentation. UI layout is a complex design problem. For this purpose, we propose the concept of UI Template, which is indexed and constrained with IMs, and is to be instantiated to become a concrete UI once the running platform is specified. Users might take part in the designing process of UI Templates by selecting the UI styles or UI design patterns and layout strategies.

4. EXTENDED OBJECT MODEL

Objects and their relationships are the main components that will appear in UI. Generally, the basic data types of object data members include numeric, character, enumeration, date, navigation and multi-medias, etc. When appearing in an UI, they must take a certain visual forms and different data types pertain to different presentation forms. Further more, object relations also have influence over UI presentation, such as UI widget layout and operations. Therefore, objects are UI presentation related, and the simple naked objects are not sufficient enough for complex UI modeling. To completely and formally depict the UI composition and behavior, new attributes and properties are needed to describe the object data members, such as range of value, data source, I/O forms for Presentation or data Acquisitions, value derivation, associations and grouping. This paper not detail described these new attributes and properties. However, Member Grouping, as an illustration, is an exception.

Normally, a data member of an object is mapped to certain IO objects in UIs. To make UI widget layout friendly and psychologically accepted by users, therefore enhancing UI usability, it is necessary to arrange and group IO objects in particular form or order. This is the so-called Member Grouping. For example, data members with the same types or closer conceptual relations might be required to be placed together and in certain order. Data members, that are grouped together, are considered as a new UI unit as a whole. In this way, the whole object is regarded as an ordered composition of UI units, forming a sequential and hierarchical arrangement of data members. All UI units are laid on an UI in certain order, for example, from right to left, from top to down or in columns.

5. INTERACTION MODEL

The IM is used to directly describe the UI composition and relationships. Its elements include Data Objects and Data Collections, Query Objects, Controlling Objects, External Entities or Components, Actors or Roles, UI Functions and UI Navigations, Data Files and Print Documents.

In our symbol system, Data Object is represented as a rectangle, in which the name, class and class members are included. Rounded rectangle and beveled rectangle are also symbols for data objects, the former as the object for forming query information, therefore being called Query Object; and the later for forming controlling information to take part in the operation of use-cases, therefore being called Control Object.

Data Collection is represented with 2 overlapped rectangles. It is an important object that might appear in UIs, therefore needing special treatment because of its wide range of usages in UI construction.

Data Object and Data Collection have the attribute for their data source. If the data source comes from a Database table, special code is needed for its implementation.

Rectangle with two vertical bars is the symbol for other UIs. Dotted arrow directed to it means the relations of UI Navigation. Print Document is a special kind of UIs in modeling and presentation with data output only in them.

Rectangle with one or more small circles connected to it represents an External Entity or Component, where the circles are its interfaces.

Use-cases directly connected to Actor are correspondent to menu items or command buttons in UI.

Object relationships are the functional associations of UI objects, which include the use of objects or interfaces, method or function calling, UI navigation, message sending or event triggering, data transmission, and object's taking-part-in relations.

UI objects in an IM can be grouped together. A Group can be viewed as an abstract entity for UI composition or an UI object. Two kinds of group can be defined. Normal group (represented as dotted-line rectangle) is the group of which each UI object will exclusively possess its own presentation space; Overlapped group (represented as bold-dotted-line rectangle) is the group that all of its UI objects will be presented overlapped and commonly in a presentation space. Events or functions can be attached to a group. They are represented as small circles.

6. PRSENATION MODEL

The presentation model deals with the specification of visual forms of UI objects and their layout as a whole.

An UI is normally presented within a rectangle area of screen. Therefore, an UI is considered as a Presentation Object that is presented in a Presentation Space.

To establish a presentation model of a complex UI with multiple constituents, the space needs to be divided into smaller rectangle areas to accommodate certain presentation objects. In this way, a presentation model of UI is superficially composed of multiple smaller presentation spaces and a set of presentation objects.

There are many ways to divide a presentation space. For ease of visual manipulation and considering being still powerful enough to deal with UI presentation complexities, the presentation space is specifically divided in the way that the dividing of the main space is consisted of a series steps, each dividing step concerns only with one single rectangle area, and is performed either vertically or horizontally into two or more disjointed rectangle areas. Thus, starting as the root, the main space is divided and organized in a tree structure, of which the leaf nodes are the resultant possible presentation spaces for accommodating presentation objects.

A Presentation Unit, or PU for short, is the combination of a leaf node or a resultant presentation space and the related presentation objects that will be presented in the space.

An UI Template contains a main presentation space that is divided and a set of PUs, where the presentation properties or details are specified. Actually, an UI Template is a set of defined relations between presentation objects and leaf nodes of a divided main presentation space, with presentation properties set and global layout specified.

Since all visible UI objects in IM require some forms of visual presentation, they will act as the Presentation Objects in PUs. Therefore, once a main presentation space is defined and divided, and the UI objects of an IM are assigned to the resultant

presentation space, an UI Template could be constructed accordingly.

To fully describe an UI in enough presentation details, more properties are needed for UI objects to be presented properly and satisfactorily. Besides presentation properties such as foreground and background colors, fonts, margins, line widths, etc., the main concerns come from the consideration of presentation constituents and their layout strategies. These properties are different for different UI objects.

Actually, when certain category of UI objects and different presentation is concerned, the similarities and differences in operation, presentation constituents and properties will help to form the concepts of UI Design Patterns. The detailed presentation constituents and their layout of each PU are determined by designer's preference and customization of UI Design Patterns for an UI object.

7. AN EXAMPLE

This example will show the grouping concept in modeling overlapped presentation and the ability in dealing with complex UI construction. Figure 2 is a practical UI in a net-loss management system (now only in Chinese version) of electrical power network. It is generated from its IM present in Figure 3 for practical application in Visual Basic. EOM of example is omitted in this paper as the limited length.

Figure 2 shows that the UI is consisted of 5 parts: 1) the Tree on the left; 2) the Page at the bottom; 3) the Graph in the right-middle for displaying graphics; 4) 6 Command Buttons on the top-left; and 5) 2 Radio Buttons and 2 Colored Blocks on the top-right. In Figure 3, they correspond respectively to the UI objects of: 1) the collection Tree; 2) the overlapped Group with bold-line rectangle and its contents; 3) the Graph component that is implemented in DCOM; 4) the 6 visible use-cases; and 5) the Control object. There is also an un-visible object to indicate the WorkingStatus.

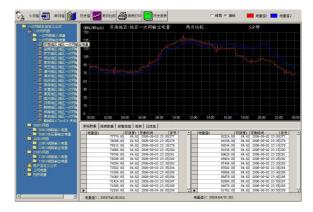


Figure 2. A practical UI generated for a net-loss management system of electrical power network

The overlapped Group represents a set of UI objects that are going to be presented and overlay in a presentation space. It is used to abstract the phenomena of overlapped presentation. In Figure 3, the Group represents an overlapped presentation of 5 sub-groups or normal groups with doted-line rectangles and their

contents. The small circles attached to each group represent the group's events or functions. In Figure 3, they designate the click events. The Control object contains 2 Radio Buttons. Their values are defined as mutual exclusive.

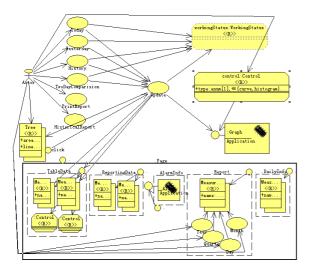


Figure 3. The IM of an UI for a net-loss management system

The operational relationships in Figure 3 are as follows: 1) Once a node of the Tree is clicked, the Update un-visible use-case is invoked. It gets the data from the node and the WorkingStatus. The data is used to invoke the Group's refresh function for TableData to update its display. Then, the related data is got and sent to the Graph component for display; 2) Upon clicking on the Group, the specific page's refresh function is invoked to update the related display; 3) Upon clicking the 4 of the 6 visible use-cases, WorkingStatus will be modified and the Update un-visible use-case be invoked again accordingly; and 4) Upon clicking the radio buttons of the Control object, their mutual exclusive relation will cause them to change values accordingly, and then the changed value be sent to the Graph component to alter the display types from histogram to curve, and vice versa.

The above relationships are all implied in the AUI of Figure 2. But, the relationship details are contained in the internal logical description of the related use-cases, which has not yet been implemented. During this experiment, the operational code is manually supplied after generating the UI framework and the related code.

To generate the desired UI of Figure 2, use method introduces in section 6, the UI template will be first divided five smaller presentation spaces according to user requirements. Next, the UI template is customized as the assignment of: 1) the UI design pattern of Tree with collection to the collection Tree; 2) the UI design pattern of Page to the Group; 3) the Graph component as it is designed; 4) Picture-Buttons to the use-cases; and 5) the UI design pattern of control object in free-form to the Control object.

Then, the presentation model of example will built, it not further description in this paper.

8. CONCLUSION

The model-based user interface development approaches have been proposed have not enough ability in deal with complexity of user interface in practical software design. This paper proposes a model driven development approach of complex user interface. The uniqueness of this research is a direct and leveled abstractive attack on complex user interface composition or description, rather than the elicitation from task models.

We have finished a UI model establishment tool that we call it AUI. Like Rose, AUI is an integrated model edit environment. Using AUI, designer can establish the abstract model of every level through the method above (Figure 3 is a hard copy of models that created in this tool.). And, now we have finished the transformation from abstract model to Visual Basic and ASP.NET.

The building of the environment and research experiment has shown that approach proposed in this paper is appropriate, feasible, effective, powerful and easily acceptable in complex and highly usable UI modeling and construction.

9. REFERENCES

- [1] Paulo Pinheiro da Silva, User interface declarative models and development environments: a survey, In Proceedings of DSVIS 2000, 2000,pp. 207–226.
- [2] Griffiths, Tonya, Barclay and Peter J, et al, Teallach: A model-based user interface development environment for object databases, Interacting with Computers, vol. 1, 2001, pp. 31-68.
- [3] K. Mitchell, J. Kennedy, P. Barclay, A Framework for User Interfaces to Databases (DRIVE), in Proceeding of AVI, ACM Press. 1996.
- [4] Angel Puerta and Jacob Eisenstein, Towards a General Computational Framework for Model-Based Interface Development Systems (MOBI-D), in Proceeding of ACM IUI 1999, Redondo Beach CA USA
- [5] Nunes, N.J. and J.F.e. Cunha, Wisdom A UML based architecture for interactive systems, in Proceeding of DSV-IS, Limerick, Ireland, 2000, pp.191-205.
- [6] Giulio Mori, Fabio Paterno and Carmen Santoro, Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions, IEEE Transactions on Software Engineering, 30(8), 2004, pp.1-14.
- [7] Pedro J. Molina, Santiago Meliá and Oscar Pastor, JUST-UI: A User Interface Specification Mode, in Proceedings of CADUI 2002, Valenciennes, France, 2002, pp.63-74.
- [8] Krzysztof Gajos and Daniel S. Weld, SUPPLE: Automatically Generating User Interfaces, in Proceedings of IUI'04, Funchal, Portugal, 2004, pp. 83-100.