User Interface Design Model

Lu Xudong
School of Computer Science and Technology, Shandong University, China
Dongxul@sdu.edu.cn

Wan Jiancheng
School of Computer Science and Technology, Shandong University, China
Wanjch@sdu.edu.cn

Abstract

To promote the model-based software engineering development of user interfaces, this paper proposes an EIP model that can design user interfaces in abstract and can support automatic generation of user interfaces. In the model, functionality and composition are taken as the direct descriptive objects of engineering modeling, making user interface construction more intuitive and easily acceptable; and a layered approach is adopted for both abstraction and presentation of user interface description, making the modeling powerful enough in dealing with complex user interfaces. In the research, internal data models and their related user interface design patterns are isolated as the user interface modeling components, making presentation and layout more easily constructed. Based on EIP and tools, the automatic generation of user interfaces is achieved.

1. Introduction

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. Aimed uniquely at the UI design and development, these approaches try to give a complete solution of UI design methodologies and environments to capture the analysis of interactive tasks and to generate UIs automatically.

However, today, various MB-UIDE researches make use of various declarative models to capture UI requirements, composition and construction; no consensus has been reached and no practically sound method has really emerged [2]. There are few MB-UIDE models and tools being fully developed and powerful enough to be recognized for commercial and industrial acceptance [3].

In the author's view, the reason of this situation is there exist some problems in the current researches on UI modeling. They are summarized as following.

1) Object data members, their types, domain of values, relationships, etc., all have some contribution to the UI constitution, layout and operational restrictions. But these types of information have not yet been fully emphasized and discussed in previous UI modeling.

2) Current UI modeling techniques lack an intuitive mechanism for directly describing UI composition and presentation with related abstract concept.

3) The task oriented UI design analysis and composition could not fully reflect users' synthetic and multiple views toward UIs.

4) The presentation models in previous approaches only outline the layout of user interfaces, not enough ability in refinement and decoration of UI.

5) The complexity of the models themselves has affected the usability of these approaches [4].

On considering above related problems, and to promote the advance of Model-Based design of UIs, we propose a brand new UI model called EIP. The main contributions of this research are:

1) Extended application models with new attributes and properties, making UI modeling can get more information to deal with UI complexities.

2) UI functionality, composition, presentation and their details, such as event-response functions and components used in UI, are taken as the direct descriptive objects of UI modeling, rather than the elicited results of other models, making the modeling more intuitive, easily acceptable and detailed enough.

3) To reduce the workload in detail design, UI design patterns, which encapsulate details of construction and implementation, as modeling elements are provided.

0-7695-2909-7/07 $25.00 © 2007 IEEE
DOI 10.1109/SNPD.2007.287

IEEE computer society

4) In presentation models, UI template used to specifies the UI overall layout and assignment of all presentation objects and components will be present in UI. Through the selection of layout strategies and customization of parameters, refine and decorate UI will achieved.

5) With the above peculiarities and help with tools we finished, high usable user interface development is becoming a simple definition and customization without involving too much details and workload.

The contents of following sections are: Section 2 discusses the related works; Section 3 outline the framework of EIP; Section 4, 5 and 6 describe the Extended Object Model, Interaction Model and Presentation Model respectively, especially focus on how Interaction model direct describe the composition and behavior of UI; and the last section ends the paper.

2. Related works

Reported researches of MB-UIDE includes Drive [5], MOBI-D[6], Wisdom[4], Adept[7], Teresa[8], Teallach[3], JUST-UI[9], UMLi[10], SUPPLE[11], etc. There also have some researches of UI description languages such as UIML [12], XIML [13] and UsiXML, but they are only languages for describing UI structures in XML format, and therefore fall in the category of UI programming at most, with the advantages of bridging among different UI programming environments.

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).

In most approaches, task model is widely used. A goal-oriented task hierarchy, with its leaf nodes termed primitive tasks, is normally been used to represent interaction or action tasks. And, to describe UIs in enough details to facilitate the UI generation, task model is enhanced to incorporate notations such as Task Types and Temporal Operators [8]. For example, in ConcurTaskTree of TERESA, Task Types includes User Task, Application Task, Interaction Task and Abstract Task; Temporal Operators includes Enabling, Disabling, Interruption, Choice, Iteration, etc.

Kee summarized the criticism [14] toward task models, such as: over emphasis on analysis, rather than on design; limited scope of application in the design cycle; under specification of application domain; and not relating explicitly to system development of software engineering methods.

We have also noticed the instability of task models in UI modeling. In most MB-UIDE researches using task models, the decomposition of task and subtask will eventually end in the so-called primary tasks. Primary tasks normally are related with input or output of simple data types. In this way, if the content of an object is going to be changed, the task model will be changed accordingly too. By using IM (Interaction Model) but task model to formulate UI requirements in EIP, UI is composed of gross granularity and relatively big objects, rather than small data items. This will keep UI modeling as more abstract and stable as possible, so long as the overall objects and their relationship do not change.

The other reason for not use task or dialog model in our approach is, 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. The task-oriented design with temporal constraint is the old fashioned command-line design, which is suitable only for that kind of UIs.

Unlike task-oriented modeling, researches in [9][10] directly model UI composition. In UMLi [10], the logical composition of UIs is described as extended activity diagrams rather than incorporating a completely new notation into UML. To this purpose, it provides user interface diagram, a specialized visualization for APMs. It also provides notations for a set of task macros that are usually observed by users. In [9], JUST-UI's direct UI modeling is supported by conceptual or more precisely interactive pattern that might be observed in UI construction.

Though researches in [9][10] and EIP all deal with UI composition directly, the layered abstract of UI modeling and presentation modeling is not supported in [9][10].

3. General Description of EIP

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 traditional object model through adding new attributes and relations for describing data members and their behavioral relationships.

Based on EOM, IM acts as a mechanism for direct description of UI. It includes functions, abstract objects,

components, external UIs, and their inter-operational 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. The UI Template is actually another form of APM, which is indexed and constrained with IMs, and is to be instantiated to become a concrete UI once the running platform is specified.

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.

The following is a brief explanation to EOM, concerning only to properties related to UI construction. Omitted attributes for class description including Associations, Measure of Unit, Default Value, Member Label and Operation Restrictions. The relationships of class derivation and class views are also omitted.

Data Sources, Range of Value and I/O Forms for Presentation. Data Source determines the origin to get values for an object data member in UI. Range of Value restricts the acceptable values that a data member might take. For UI I/O objects or controllers, this gives criteria for possible input checking for errors. I/O Forms for Presentation suggests the visual form or interaction style that a data member might take. Usually, different data types take different forms to present or acquire values, and therefore use different I/O objects to fulfill their operations.

Derived Relation, Behavior Association and Member Grouping. Some data objects may take values only from other objects' values. They do not accept values directly from input sources. When relating objects' values changed, the data objects will change their values accordingly. This is known as the Derived Relation of objects, which relation recognizes a type of UI object relations and behaviors. If an object's value is derived from other objects, it does not accept any I/O operation. And any change of relating objects' values will trigger the evaluation operation of the derived object's value. Therefore, only static text or label is necessary for its UI presentation, and the evaluation operation needs special structure.

Behavior Association is the behavior or controlling concerned relations of IO objects. If one IO object's operation might affect or trigger other objects' behavior, it is said that there exists some behavior association among data members. For example, the enabling relation of UI objects is such an association.

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.

Example. The EOM description of collection CollectionOfOrder's object in the example of stock replenish and assessment is as follows:

BuyerID: String20, UI:Edit, Label:BuyerID, association with CollectionOfDetails.BuyerID;
BuyDate: Date, UI:Datetime, Label:BuyDate;
SupplierID: String20, UI:Combox, label:SupplierID;
SupplierName: String20, UI:Edit; label:SupplierName;
Purchaser: String20, UI:Combox, Label:Purchaser;
DueDate: Date, UI:Datetime, label:DueDate;

The EOM description of collection CollectionOfDetails' object of is as follows:

buyerID: String20, UI:Edit, Label:buyerID;
goodsID: String20, UI:Edit, Label:goodsID;
goodsName: String20, UI:Edit; Label:goodsName;
goodsUnit: String8, UI:Combox, Label:goodsUnit;
goodsPrice: Num12.2, UI:Edit, Label:goodsPrice;
goodsNumber: Num8, UI:Spinner, Label:goodsNumber;

5. Interaction Model

The IM is used to directly describe the UI composition and relationships. In the following, the main concepts of IM will be explained, and example will be given to show UI modeling by using IM.

Data Collection, Query Object, Control Object and Component. Data Collection is abstracted as a collection of objects, which might be instances of a concrete or a view of EOM class. Because of its very

structure and UI presentation as well as the functionality, data collection is necessary for UI modeling in concepts, because many UI operations are concerned with various collections.

All data objects and data collections have the standard or pre-defined abstract behaviors in the form of various methods. For example, a data collection has the methods for querying, sorting, adding and deleting objects. All visible data objects and data collections have standard UI event-response methods.

Query is a common and widely used operation; therefore, the forming of query conditions is an important operation in UIs. Because of its special behavior and presentation, Query Object is specified for the forming of query conditions in UI modeling. Though it is called an object, it is not the object for the information to be produced, but an object that takes part in other operations. Its data members come from the related EOM definition, but its presentation and behavior is quite different from ordinary data objects.

Control Object is an object that taking-part-in other processing. It usually acts as constituents of pre-condition, or parameters for processing, or state variables for controlling other components or objects. Its functionality is to transfer or update values to take part in other UI behaviors or processing.

Component or External Entity is a pre-fabricated software unit and might comes from outside. Usually it represents a system resource. Its adoption in UIs is to show special functional services resource and to support component based UI development. This is especially important for an UI model to be practically usable and powerful.

Associations. The association relations of objects also contribute to the complexity of UIs, since they complicate the interaction among objects and are inevitable in UI modeling. In general, there exist such different categories of associations such as single object or single collection with one or multiple associations. Association is a kind of object relations, with the related semantic constraints specified. The existence of the association will affect the UI code generation, making the association meaningful.

UI Functions and UI Navigations. UI Functions and UI Navigations are the results of preliminary system analysis. In IM, they are described in details with the help of various objects. UI Functions act on and therefore are associated with various objects. Method calling, message sending, event triggering, data transmission and UI navigation make the functional association more clear and evident. UI Functions in IM are represented as visible use-cases.

There are two kinds of UI Navigation. They are modal or non-modal forms. Parameters can be transferred into or out of UIs for UI navigation. These all will contribute to the automatic UI generation, especially the generation of UI code frameworks.

Use-Case. Use-case represents an operation that contributes to the overall functionalities of UIs. Usually, various objects are "assembled" together by a use-case. There also exists the difference as visible or non-visible use-case. The visible ones are correspondent to menu items or command buttons or hype-links, which will serve as the UI functional services. The non-visible ones are correspondent to internal services. Like all other visible object, visible use-cases can also be Enabled or Dis-Enabled with appropriate operations.

The semantics of relations of use-cases with other elements is operational in nature, meaning that the associated objects with the relations will take part in the behavioral operations of the use-case in certain order. The desired operations that will be involved in the processing of use-case are specified as labels on the relations. In this way, the symbols for a use-case and its relations are the representation of the "algorithmic architecture" of the use-case.

The order that each object will take part in the operations of a use-case could be specified in the relations with numbered labels. For a use-case, numbered labels with the same number means the concurrency in operations. In consideration of relations with tasks, this is similar to the Task Model for temporal logic specification.

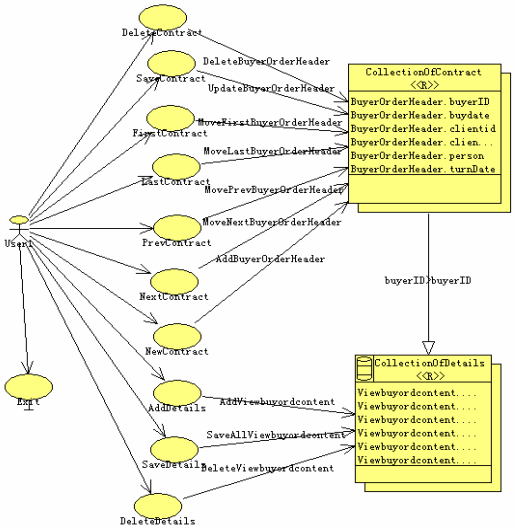

Figure. 1. The interaction model for PurchasingContract

Example. Figure 1 is the IM for an UI in the stock replenish and assessment example. The conceptual composition of the UI is described in it, including 2 data collections and 10 use-cases, as well as their

operational relations. Of the 10 use-cases, 7 are operating on CollectionOfOrder, 3 on CollectionOfDetails, and 1 for exiting of the UI with an "earth" symbol attached. There exists a master-slave association constraint between the 2 collections. The collection CollectionOfOrder is the master and CollectionOfDetails is the slave, with the constraint that the master's "buyerID" equals to the slave's "buyerID". The existence of this constraint requires that when the focused object of the master is changed, the displayed content of the slave will be updated accordingly by maintaining the equal values of two buyerIDs.

6. Presentation 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.

The considered properties for presentation also include layout strategies, such as Left-Right, Top-Down, Vertical Space, and Horizontal Space.

In fact, once an IM is constructed, and a presentation space is divided, and mapping relations of UI objects with the leaf nodes are assigned, and each PU's presentation form and properties are selected and assigned manually, an UI Template is formalized. Again, once the destination platform is selected for the UI, the outmost CPM might be automatically generated from the UI Template as well as its related IM. But limited the length of paper, we omit description of UI design patterns and code generation.

Example. The actual dividing of an UI presentation space and their mapping associations with UI objects in IM, as well as user's presentation preferences and setting of various properties, are all manually determined at the design time of UI Template under the index of the related IM. Figure 2 is the UI Template for IM in Figure 1. Of course, the presentation of UI template is only a sketch map for the intended UI to be generated.

In the UI template of Figure 2, various constituents of the IM are assigned and placed on the sketch map of UI. Firstly, the whole rectangle space is divided vertically into 4 areas, of which its second area is further divided horizontally into 2 areas. Secondly, each area is assigned with one or more object from the IM. In the example, from top to down and from left to right, the areas or PUs are assigned in the sequence of: CollectionOfOrder, 4 use-cases for record movements of CollectionOfOrder, 3 use-cases for operations of

CollectionOfOrder as adding and saving and deleting, CollectionOfDetails, 3 use-cases for operations of CollectionOfDetails and 1 use-cases for exiting. The presentation preferences are manually set for each PU as: Free-form for CollectionOfOrder, Grid for CollectionOfDetails, and character-command-buttons for all visible use-cases.

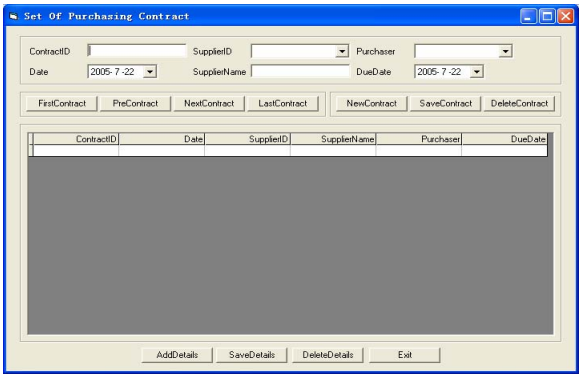

Figure 2. The presentation model for PurchasingContract

The above assignment will eventually form a description of UI layout, which is described by using the recursive tree structure and organized in the form of inter-leaved Vertical(…) and Horizontal(…) constructs.

Moreover, some decorate information, such as logo and pictures also can assign the presentation space. But this is not show in example.

7. Conclusion and Future Works

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 (All figures about example in this paper are 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 EIP is appropriate, feasible, effective, powerful and easily acceptable in complex and highly usable UI modeling and construction, especially for rapid UI prototyping and customization.

The future work will be focused on the consummation and applications of the model, the enrichment of the user interface design patterns and the standardization. Especially we need to add the formalized description of the operations and the arithmetic logics, so as to achieve the general generating of the UI codes.

8. 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] Jean Vanderdonckt, “A MDA-Compliant Environment for Developing User Interfaces of Information Systems”, In Proceedings of 17th Conf. on Advanced Information Systems Engineering, 2005,pp.16-31.

[3] 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.

[4] Nunes, N.J. and J.F.e. Cunha, “Wisdom – A UML based architecture for interactive systems”. In Proceedings DSV-IS, 2000, pp.191-205.

[5] K. Mitchell, J. Kennedy, P. Barclay, “A Framework for User Interfaces to Databases (DRIVE)”, in Proc. AVI, ACM Press, 1996.

[6] Angel Puerta and Jacob Eisenstein, “Towards a General Computational Framework for Model-Based Interface Development Systems (MOBI-D)”, in ACM IUI 1999, Redondo Beach CA USA, 1999.

[7] P. Johnson, H. Johnson, S. Wilson, “Rapid Prototyping of User Interfaces Driven by Task Models (Adept)”, Scenario-Based Design, (Carroll, J. ed). John Wiley & Son, 1995, pp. 209-246.

[8] Giulio Mori, Fabio Paterno, Carmen Santoro, “ CTTE: Support for Developing and Analyzing Task Models for Interactive System Design”, IEEE Transactions on Software Engineering, Vol.28, 2002,pp. 1-17.

[9] Pedro J. Molina, Santiago Meliá, and Oscar Pastor, “JUST-UI: A User Interface. Specification Model”, in proc. Computer Aided. Design of User Interfaces, CADUI'2002, Les Valenciens, France, 2002,pp. 323-334.

[10] PP da Silva and NW Paton, “User Interface Modeling in UMLi”, IEEE SOFTWARE, 20(4), 2003, pp. 62-69.

[11] Krzysztof Gajos and Daniel S. Weld, “SUPPLE: Automatically Generating User Interfaces”, In Proceedings of IUI, 2004, pp. 83-100.

[12] Mir Farooq, A., Abrams, M., “Simplifying Construction of Multi-Platform User Interfaces using UIM”. In European Conference UIML, 2001.

[13] Puerta, A., Eisenstein, J.. “XIML: A Common Representation for Interaction Data:, In 7th International Conference on Intelligent User Interfaces, 2002, pp. 214-215.

[14] Kee Yong Lim, “Structured Task Analysis: An Instantiation of the MUSE Method for Usability Engineering”, Interacting With Computers, Vol.8, 1996, pp.31-50.