A Complex Interface Modeling Approach based on Design Pattern

Teng Gao, Qingzhong Li, Shuaiqiang Wang  
School of Computer Science and Technology  
Shandong University  
Jinan, China  
E-mail:gaotenga@163.com, lqz@sdu.edu.cn, shqiang.wang@gmail.com

Abstract—User interface modeling methods based on design pattern and code generation methods are hot points in software engineering research field in recent years. However, there is no unified standard description until now. As a result, it leads to the great insufficiency in using nested component of multiple interface design patterns to construct complex user interface; meanwhile it cannot meet the need of displaying the content of a hierarchical structure in the limited area. In order to enhance the ability of the interface design pattern to support complex interface description, a complex interface modeling method is proposed in this paper. Based on abstracting the interface basic elements, standardized description of interface design pattern is realized; complex interface modeling and target code automatic generation are finally realized by customizing interface basic elements. Application research shows that this method can greatly support complex interface design and realization, and enhance the efficiency of user interface development.

Keywords—Complex Interface; Design Pattern; Interface Basic Element; Code Generation

I. INTRODUCTION

The effect of interface design is an important factor in evaluating the software system. All kinds of research on software reuse and interface design pattern provide academic and practical bases for interface engineering research. Now, interface engineering research mainly focuses on two aspects, which are interface exploitation and formal interface design description based on models. They construct interface design standardization and exploitation process from different aspects and layers. Single formal interface design description cannot realize systemic and engineering interface exploitation[1]. The model-based interface exploitation methods lack of the effective reuse of normal design knowledge[2-3], and cannot work efficiently, thus restricting the further development of interface engineering.

An abstract description model for user interface which is based on concept model is proposed by JUST/UI[4-5]. It analyzes and expresses UI concept requirement with an abstract description method. However, details of its realization are not considered; it is difficult to acquire the information related to the interface display. The model cannot show the interface prototype to users; nor can it show the way how an interface is generated by the model. As a result, it cannot better support users to participate in interface designing in the way that seeing is getting. Task model[6-7] is a domain-related model. As for the impact of object structure and relation on interface complexity, an intuitional and comprehensive study is in need. As a result of it, the model description capacity is reduced.

FUSE[8] proposes a highly synthesis development environment based on tools, which supports processes of the entire user interface development. But FUSE adopts formal concept description methods which is not easy to master and realize, as a result, it is hard to use. Furthermore, model-based interface projects lack of reusable knowledge objects[9-10] of interface design that covers all kinds of models, so it is hard to realize effective code generation. All kinds of concept pattern, interface templates and interacting pattern can only support certain phases of pattern development. What’s more, they lack of effective integration of model-based interface development. These research only thoroughly normalizes interface and its behavior, and doesn’t take into account normalizing interface design knowledge which can be reusable.

In a word, the current interface design pattern lacks of unified standardized description, which leads to the great insufficiency in using nested component of multiple interface design pattern to construct complex user interface, and also could not meet the need of displaying the content with a hierarchical structure in the limited area. Building interface design knowledge and an effective integration mechanism of model-based interface development become an urgent problem in interface engineering development.

In order to enhance the ability of the interface design pattern to support complex interface description, a complex interface modeling method is proposed in this paper. According to abstracting the interface basic elements, standardized description of interface design pattern is realized, and according to customizing interface basic elements, complex interface modeling and target code automatic generation are finally realized. The efficiency and quality of interface development are improved.

II. COMPLEX INTERFACE

A. Definition of Complex Interface

A complex interface can be seen as a display area, which can be partitioned to a series of sub-areas. The relation between complex interfaces and sub-interfaces can be seen
as a tree that is composed of display area nodes. Every node can be seen as a display area, and the root node expresses the whole complex interface. The child nodes set of every node expresses the partition of display area that the node describes. The leaf node expresses the smallest display area that can’t be partitioned, and it is the container of the interface design pattern. According to assigning corresponding interface design pattern to the leaf nodes of the display area, the design of the sub-area can be completed. It is shown in Fig. 1. The nesting and combination of the interface design pattern can be seen as combination of the leaf node areas.

B. The Theory of Interface Design Pattern

According to the content and structure of interfaces display, interface complexity can be classified as follows:

1) The content of the interface is either simple structure or complex hierarchical structure;
2) The content of the interface is either independent or related with others.

Interface design pattern can be classified as follow:

1) Simple pattern.
   Independent interfaces with plane structure, such as list display and tree display, can be abstracted as a simple pattern by existing interface design patterns. In this pattern, display objects are common objects, and also are basic objects. Properties of objects are basic types and lack of nested relations. After assigning the objects to interface, visual object elements are directly displayed in certain form in entire interface.

2) Simple combination pattern.
   Interface with a plane structure and whose content is relevant or needs to display together, such as button groups, summarization and details, can be abstracted as simple combination pattern. In this pattern, display objects are common combination objects, which are composed of several common objects, and they have only one layer of nesting. After assigning it to a particular area, this area can be divided into some sub-areas where sub-objects can be assigned.

3) Overlapped combination pattern.
   Multi-hierarchy or multi-contents that need to display in the same area, such as multi-level navigation menus and multi-layer property pages, can be abstracted as a pattern, overlapped combination pattern. In this pattern, display objects are overlapped combination objects, which are composed of common objects and combination objects, or combination objects and combination objects; they have two layers of nesting at least. After assigning it to a particular area, this area is composed of some overlapping interfaces, which display one interface a time either in a cover way or in a paging way. For each interface, we can assign sub-objects to it which belong to the combination objects. After the assignment, according to the type of the assigned objects, corresponding operation can be carried out until each interface can be displayed directly as common objects.

III. INTERFACE DESIGN PATTERN

Interface design pattern is defined as follow:

Definition 1 Interface design pattern UIDP = (Data Set, Property Set, Action Set, Present Set).

Namely, an interface can be abstracted as a combination or recursive combination which is composed of four basic elements, Data Set, Property Set, Action Set and Present Set.

A. Interface Basic Elements

In this paper, a concept of composite data set is proposed to describe and operate the complex interface. It describes interface design pattern from four aspects: data, property, action and display; four model elements are introduced to describe related interface information from four aspects: the interface data information described by Data Set, related control property information in the interface design pattern described by Property Set, interacting information adopted in operations described by Action Set; the ultimate interface exhibition information described by Present Set. In addition to the above four types, every interface has an additional Flag attribute, which is used to describe interface complexity. Detailed definitions are as follows:

1) Data Set
   Data Set describes the data information that interface displays, including names, sources and types, etc.

Definition 2 Data Set = (DName, DFrom, DType).
   DName expresses data name, which is the primary key of the Data Set. DFrom expresses the origin of the data, which explains the data coming from user inputs or external storage. DType expresses data type.

2) Property Set
   For a certain data in the Data Set, Property Set describes the set of form’s properties displayed by data in interface design pattern; each property is defined as follows:

Definition 3 Property Set = (PName, DName, PType).
   PName expresses property name, which is the primary key of the control property set. DName is the foreign key, which is the data needs to be displayed. PType is the property type, which is the controls type of display objects, such as Button, List and Tree, etc.

3) Action Set
   Action Set describes the set of operation in the interface design pattern, which includes event and action. Event
describes the response of corresponding pattern objects to the operation that users make in the interface, while the action describes the operation that the pattern objects are called by external pattern objects to realize the relations of different pattern objects, that is, the relationship of different modules in the interface. The definition of operation is as follows:

**Definition 4** \( \text{ActionSet} = \langle AName, PName, ARS, RType \rangle \), \( AName \) expresses action name, which is the primary key of Action Set. \( PName \) is the foreign key, which expresses the operation that deals with a certain control. \( ARS \) expresses the result set of the operation, that is, the result produced after carrying out an operation. \( RType \) expresses the return type of the result.

4) **Present Set**

Present Set describes the ultimate display form of the interface design pattern. Interface design pattern has a variety of display ways and styles. These ways and styles differ from each other, but also have some common features. So this requires Present Set to describe all kinds of features:

**Definition 5** \( \text{Present Set} = \langle \text{Back}, PName, \text{Layout}, \text{App} \rangle \), \( \text{Back} \) expresses entire appearance of the display area. \( PName \) is the primary key, which expresses the operation dealing with a certain control. \( \text{Layout} \) expresses the layout of the control set in the display area. \( \text{App} \) expresses the appearance of the control or interface.

5) **Flag**

Flag expresses the display pattern of the interface or display area. Namely, simple pattern, simple combination pattern or overlapped combination pattern. If \( \text{Flag} = 0 \), it means that the display pattern is simple pattern; else if \( \text{Flag} = 1 \), it is simple combination pattern; else if \( \text{Flag} = 2 \), it is overlapped combination pattern.

**B. Abstract Description of Complex Interface**

According to description above, an interface can be seen as a tree that is composed of display area nodes, in which a leaf area includes one or more interface design pattern. We can use recursive expressions to make an abstract illustration.

\[
\langle \text{Interface} \rangle := \langle \text{Display Area} \rangle \quad (1) \\
\langle \text{Display Area} \rangle := \langle \text{Display Area} \rangle * + \langle \text{Association} \rangle * \quad (2) \\
\langle \text{Display Area} \rangle := \langle \text{UIDP} \rangle * \quad (3) \\
\langle \text{UIDP} \rangle := \langle \text{Data Set} \rangle * + \langle \text{Property Set} \rangle * + \langle \text{Action Set} \rangle * + \langle \text{Present Set} \rangle * \quad (4)
\]

\( \langle \text{Association} \rangle * \) describes the relationship between sub-display areas, that is, the relationship of leaf nodes in the interface tree. \( \langle \text{Association} \rangle * \) includes the following:

1) **Data Sending**

That is sending data between the display area including query condition objects or control parameter objects and the display area including visual data objects.

2) **Function Calling**

It expresses the relation between sub-interface including use case and sub-interface including data object or collection, which constructs the association between the use case and the operation data by invoking data objects or the action of collection.

3) **Participation**

It expresses input parameters of use case in the display area; control parameter objects or query condition objects generally point to the use case.

4) **Interface Navigation**

It expresses the navigation relations of the display areas.

**C. Modeling Analysis of Interface Structure**

Using the knowledge of the basic elements and abstract description of the complex interfaces, we can make a modeling analysis of the interface structure described in section II.

1) **Simple pattern**

For independent interfaces with plane structure, such as list display and tree display, we can use Expression (1), (3) and (4) to model.

Namely, according to the simple pattern, if the property \( \text{Flag} = 0 \), we can complete the modeling only by describing \( \langle \text{Data Set} \rangle, \langle \text{Property Set} \rangle, \langle \text{Action Set} \rangle, \langle \text{Present Set} \rangle \). It is shown in Fig. 2.

2) **Simple combination pattern**

Interface with a plane structure and whose content is related or needs to display together, such as button group, summarization and details, etc, has a layer of nested content. According to the features of this pattern, we can use Expressions (1), (3) and (4) to model. The model is shown in Fig. 3.

3) **Overlapped combination pattern**

As shown in Fig.4.Multi-hiberarchy or multi-contents that need to display in the same area, we can use Expressions (1), (2), (3) and (4) to model. A detailed description is as follows:
For a complex interface, Expression (1) is called:

\[ < \text{Interface} > := < \text{Display Area} > \]  

At first, the complex interface can be seen as a display area. For the entire display area, Expression (2) is called:

\[ < \text{Display Area} > := < \text{Display Area} > * + < \text{Association} > * \]  

The entire display area can be further divided into several sub-areas \( N_1, N_2, \ldots, N_m \) and the association of these areas. For a certain sub-area \( N_p \), it can be divided into smaller sub-areas.

For the display areas in the leaf nodes, Expression (3) is called:

\[ < \text{Display Area} > := < \text{UIDP} > * \]  

That is, an interface design mode is assigned to fill in the display area. Expression (4) is called:

\[ < \text{UIDP} > := < \text{Data Set} > * + < \text{Property Set} > * + < \text{Action Set} > * + < \text{Present Set} > * \]  

For the property sets in Expression (4), we can complete an area modeling process by the definitions and evaluations of property. Other sub-areas can be modeled recursively, and at last we complete the modeling of the whole complex interface.

For a complex interface like \( \text{Flag}=2 \), we can first divide it into several display areas, which have a variety of display forms, that is, Simple display, Paging display and Tab page display. These display forms are controlled by the property \( \text{App} \) in the Present Set. If \( \text{App}=0 \), it expresses that the area is displayed in a simple display way. If \( \text{App}=1 \), it expresses that the area is displayed in a paging display way. If \( \text{App}=2 \), it expresses that the area is displayed in a Tab page display way. Each design of the display area is a combination of the established four elements.

IV. A CASE STUDY

In this section, a case of employee management system is shown. In the case, modeling is achieved by the mentioned modeling methods of the interface, and a code based on ASP.NET is finally generated.

AUI[11] is applied to design Data Set, Property Set, Action Set and Present Set. This case states an Employee class in data view to model Data Set and Property Set describing the Employee information. As shown in Fig. 5, in this case, Employee ID, Department ID, Employee Name, Sex, Birthday and Employee Position are respectively defined, in which the Employee ID is the primary key.

Fig. 6 explains a situation of Action Set, which shows interaction between different areas of the interface or interaction between the interfaces and users. According to complex interface design pattern mentioned above, Fig. 6 is divided into four parts: the top layer of the entire process is the users who operate the entire program. The second layer is use case group objects which includes Query, Add, Edit, Update and Delete. It models the area that provides services in the interface. The third layer is a collection object, which models the area that displays several pieces of Employee information in the interface. The last layer is detailed information expressing the overlapped combination objects. We make property \( \text{App} \) of Present Set equal 2, that is, adopting Tab page display form. Tab page is composed of Resume, Reward, Punishment, Paper, Project and Publication; each page has its own data and display forms.

Figure 4. Overlapped combination pattern

Figure 5. Employee class
The relation from the users to each use case group object is *function using*, that is, the users employ the functions mentioned above. The relation from use case group objects to the *Employee* object is *function calling*, which means that the data of the collection pattern is called by serve pattern. The relation from the *Employee* to the combination pattern is *data sending*, which means that the data of *Employee* is sent to each object of combination pattern.

An interface situation designed by Fig. 6, one of display ways is shown in Fig. 7. The designers can customize the display properties of layout and appearance. Three areas in the interface template respectively expresses button display style of the use case group objects, free display style of the collection objects and *Tab page* display style of the overlapped combination objects.

After the interface modeling, the model generates code on the basis of the following information: the static information composed of data and property shown in Fig. 5, action information shown in Fig. 6 and interface display information shown in Fig. 7. User interface based on ASP.NET is shown in Fig. 8.

V. CONCLUSIONS AND FUTURE WORKS

As for complex interface design deficiencies, a complex interface modeling method based on design pattern is proposed in this paper. By customizing the interface basic elements *Data Set*, *Property Set*, *Action Set* and *Present Set*, we have overcome the deficiencies of the current design models and code generation which can not support the complex interface sufficiently, and improved the capacity of software development. At the same time, it supports the generation of the interface target code. The main task for future research is to expand the support capacity of interface basic elements. Furthermore, target code generation based on different platforms is also an important research subject.

REFERENCES
