Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 166 (2020) 51–56

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Mechatronics and Intelligent Robotics (ICMIR-2019)

Research on MVP Design Pattern Modeling Based on MDA

Dan Dan Li * [11], Xiao Yan Liu [1]

Faculty of Information Engineering and Automation,Kunming University of Science and Technology, Kunming 650000,Yunnan,China

Abstract.

In recent years, mobile devices (such as various brands of mobile phones, tablet computers, smart watches, etc.) have gained popularity among users, which has also promoted the rapid development of mobile software application market. In order to meet the needs of mobile device users for better user interaction experience, the demand of mobile software application market for user interface is becoming more and more complex, and software developers are constantly exploring and developing more popular user interface. Due to the shortage of production technology, the development of mobile user interface has some problems such as low efficiency and high cost. In order to solve these problems, this paper proposes a method of modeling PIM using IFML and UML by introducing MVP design pattern in a model-driven framework. By combining the advantages of model-driven and MVP design pattern, this method improves the granularity of PIM modeling in user interface development under model-driven framework, reduces the difficulty of model design, and promotes software reuse.

Keywords: IFML; MVP design pattern; MDA; user interface development; PIM; modeling; UML

1. Introduction

With the widespread use of mobile applications, the demand for user interfaces has also increased, which has created more and more complex user interface design issues for developers. In traditional user interface development, requirements analysis and design are often represented using documents and diagrams. When user interface requirements change during development, developers often make changes to the code, making these documents and

[1] Corresponding Author. Tel.+(86) 15687893463
*Email: 2425431718@qq.com

10.1016/j.procs.2020.02.012

charts useless. Sometimes it may be continually changed according to actual customer needs, which increases the workload and is not conducive to user interface maintenance and reuse. In view of the above problems, how to closely link requirements analysis documents, charts and codes to improve software reuse and promote user interface development efficiency has important research significance.

The Object Management Group (OMG) proposed the Model Driven Architecture (MDA)[1]. Compared to traditional code-centric software development methods, MDA uses models as an important component of software development. MDA models the software requirements analysis and design into a model, namely Platform Independent Model (PIM), and obtains the Platform Specific Model (PSM) through model transformation. PIM describes the software requirements, does not involve any issues related to the implementation of the technology, and improves the abstraction level of requirements analysis and design. The advantage of this is that no matter how the demand changes, you can get the required application software by changing the software model in time, and solve the problems caused by the diversity of the actual development technology and the untimely changes of the software requirements.On this basis, zhang tian et al. proposed to use UML to model the PIM layer of observer pattern under the framework of MDA to fully support reuse and improve modeling granularity at a higher level of abstraction[2]. Wang jian et al. proposed to use UML to model the PIM layer of MVC design pattern under the MDA framework, which effectively improved the efficiency of Web application development[3].

In order to improve the development efficiency of mobile user interface, the PIM layer of MVP design pattern is modeled using IFML and UML under MDA framework. IFML mainly models the View layer and Presenter layer of MVP design pattern visually, and UML models the business logic of Model layer.

2. Mvp Design Pattern

In the actual user interface development work, there are often similar functional requirements for different projects. The general developer's inertial thinking is to design a solution for each project. Such an approach will inevitably lead to unnecessary duplication of development work, and the development efficiency will be greatly reduced. To solve such problems, design patterns are a good choice. The design pattern divides the actual problem into modules, and each module only needs to be responsible for the tasks that belong to it. For different projects with similar functional requirements, the new project can reuse the modules in the completed project to achieve high reusability of the code. Currently in the user interface development of mobile software applications, MVP pattern (Model-View-Presenter)[4] has been widely used due to its good decoupling. In the MDA framework, the MVP design pattern is introduced into the MMA PIM layer modeling. By combining the advantages of both, the software reuse and the modeling granularity can be fully supported at a higher level of abstraction, thereby reducing the design difficulty of the model.

The MVP pattern evolved from the well-known MVC pattern (Model-View-Controller)[5]. In the basic idea, they have the same place: Model is the data layer, responsible for accessing data; View is the view layer, responsible for data display; Controller/Presenter is the logic layer, responsible for logic processing. The MVP working mode is shown in Figure 1.

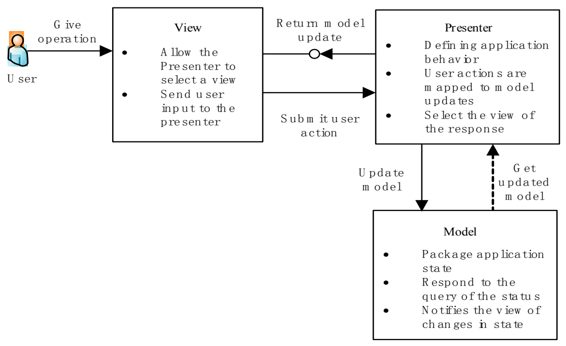
Give operation
User
View
Allow the Presenter to select a view
Send user input to the presenter
Return model update
Submit user action
Presenter
Defining application behavior
User actions are mapped to model updates
Select the view of the response
Update model
Get updated model
Model
Package application state
Respond to the query of the status
Notifies the view of changes in state

Figure.1 MVP pattern’ s architecture diagram

As shown in the MVP architecture diagram shown in Figure 1, we can clearly understand the MVP workflow. When the user gives the operation, the View submits the user operation to the Presenter. The Presenter notifies the Model to update the model. The Model feeds the updated data to the Presenter. After the Presenter processes it, the updated model is returned to the View, and the View presents the result to the user.

Compared with the MVC pattern, the MVP pattern has different aspects: 1) View does not directly use the Model in MVP, and communication between them is performed by Presenter (Controller in MVC), and all interactions occur. Inside the Presenter, in MVC View will read data directly from the Model instead of through the Controller. When the View needs to be modified, the Model is not affected by the MVP pattern, but in the MVC pattern, the Model will be affected by the View changes. 2) There is no direct connection between the Presenter and the View through a well-defined interface. When the View changes, there is no need to change the Presenter's logic. From the two different points of MVC and MVP, it can be concluded that MVP is significantly better than MVC, and the effect of decoupling and reuse is better[6].

3. Ifml Basic Elements For Design Pattern Modeling

It was proposed from the first international standard modeling language (Unifed Modeling Language or UML)[7], and there is no standard user interface modeling method. Until March 2013, OMG adopted the Interaction Flow Language (Interaction Flow Language,IFML) specification[8] serves as a new standard for user interface modeling. IFML is widely used in the field of user interface modeling due to its visualization and easy-to-operate interpretation.

IFML also provides an extension mechanism that extends the basic elements of IFML to better describe user interfaces, interactions, and other abstractions[9]. Designers can extend IFML to their actual needs. Elements that represent extensions in IFML are marked with "« »". There are many IFML modeling elements, and in the following sections of this section we will only cover the basic elements of IFML for MVP modeling.

A view container (ViewContainer) is the basic interface element of an IFML model diagram, such as a window in practice, a web page, or a container in an abstract sense. A view container contains elements that display content and elements that interact with other ViewContainers. The top-level view container is generally composed of the IFML model, and the child view container can also be nested in the top-level view container.

A ViewComponent is a component of a ViewContainer that displays content or accepts user-entered interface elements. A view component can be composed of one or more view component parts (ViewComponentPart), such as input/output parameters (InParameter/OutParameter), input fields (SimpleField), data binding (DataBinding), and so on. DataBinding represents the source of the content representing the domain model object. It can specify the characteristics of the data type, select the criteria for the instance, and publish the related properties.

A ViewContainer or a ViewComponent can be associated with an Event, which can be generated by normal or abnormal termination of user interaction, system events, or actions. An event can trigger an action, and the event is connected to the action through a navigation flow. The action itself is connected to the target view component or the attempt container by a navigation flow through an event indicating the end of the action. An action may trigger an event, called an ActionEvent, to indicate normal or abnormal termination of the action.

When the event is triggered, the navigation flow (NavigationFlow) is activated, which connects the viewcontainer, viewcomponent, ViewComponentPart, or Action with events of other viewcontainer, viewcomponent, ViewComponentPart, or Action.

The parameter binding group (ParameterBindingGroup) is associated with the navigation stream and represents the input and output dependencies between the ViewContainer and the ViewComponent or between the ViewContainer/ViewComponent and the Action.

4. MVP Design Pattern PIM Layer Modeling Design

PIM is a platform-independent model in MDA, which is mainly used to describe the business requirements and functional structure of the system. PIM has a high level of abstraction, independent of the implementation platform and any technology actually employed. For similar functional requirements owned by different projects, low-level application models and actual code can inherit and reuse all definitions in PIM. In the PIM modeling of the user interface, the MVP pattern with better decoupling and reuse characteristics is introduced, which makes the modeling more modular and higher granularity.

IFML has visual and easy-to-operate interpretation advantages in user interface modeling. We use IFML to model the View layer and Presenter layer in MVP pattern, and use UML to model the Model layer. Combining IFML and UML to model the MVP pattern makes the modeling work more convenient and easier to understand. Next, we use IFML and UML to model the three MVP modules separately:

1) Model layer: The model layer uses UML to model domain objects and business logic;

2) View layer: The view layer uses IFML basic elements such as ViewContainer, ViewComponent, Action, Event, and IFML extension elements to model it;

3) Presenter layer: The presenter layer uses the ViewContainer, Action to model it;

4) Back-end service: The backend service models it using ViewContainer, Action, and ViewComponent. A block diagram depicting the modeling is shown in Figure 2.

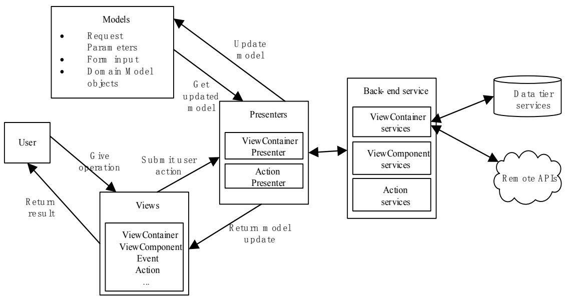

Figure.2 PIM layer model of MVP pattern

5. Examples

To illustrate the specific modeling method, we combine an example and use the modeling method of Section 3 to design a mobile application user interface based on MVP. This example describes the logic of a news search, as described below:

1) Enter the news search user interface and ask for the keyword and date. The date consists of two parts: the start date and the end date;

2) Click the search button to submit a search request;

3) If the search is successful, the page jumps to the desired news page;

4) If the search fails, the page jumps to the error message page.

Next, we use IFML and UML to model the three modules of MVP separately, as described below.

View layer modeling: This layer visually models news search using IFML basic elements and extension elements. SearchNews ViewContainer reaches the NewsFound ViewContainer or the Error Nitification ViewComponent via the SearchNews Action. The SearchNews ViewContainer includes the "«Form»" extended NewsSearch ViewComponent, and the NewsFound ViewContainer includes the "«List»" extended NewsList ViewComponent. As shown in Figure 3.

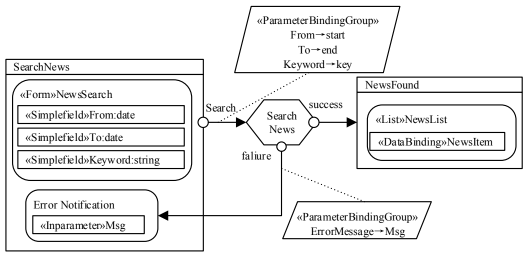

Figure 3 View layer modeling

Model layer: This layer is modeled as a SearchNews bean and uses the SearchNews service as shown in Figure 4.

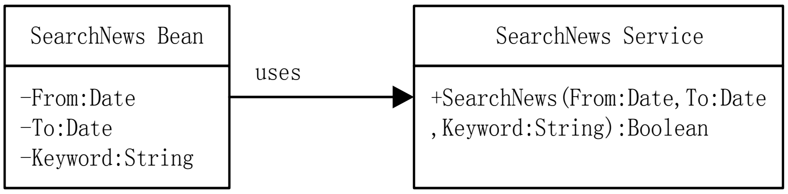

Figure.4 Model layer modeling

Presenter layer: This layer modeling includes SerchNews ViewContainer Presenter, SerchNews Action Presenter and NewsFound ViewContainer Presenter. As shown in Figure 5.

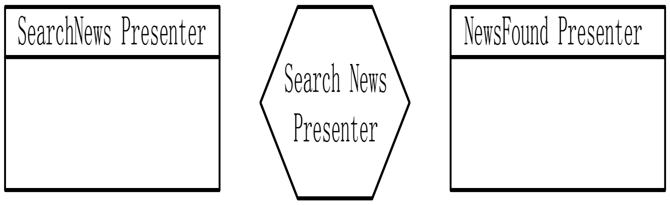

Figure.5 Presenter layer modeling

Back-end service: In this example, the backend service is modeled as SearchNews ViewContainer service, NewsFound ViewContainer service, NewsList ViewComponent service, SearchNews Action service, NewsSearch ViewComponent service. As shown in Figure 6.

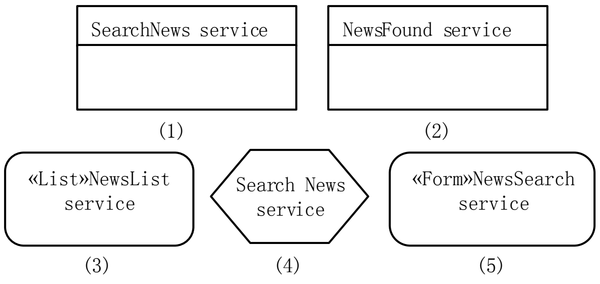

Figure 6 Back-end service modeling

6. Conclusion

This paper proposes a method to introduce the MVP design pattern into MDA. The IEML is used to model the View layer and the Presenter layer of the MVP design pattern, and the Model layer is modeled using UML. Finally, an example is given to illustrate the specific modeling method. Combining the advantages of both MVP design mode and MDA, the PIM modeling granularity and the reusability of the software model are improved, and the problem of low efficiency and high overhead of traditional user interface development is solved. Modeling the MVP design pattern is the first step in our research. The next step is to design the model transformation rules from MVP PIM to the specific implementation platform and the final generation of the code.

7. References

1. Anneke Kleppe,F., Jos Warmer,S.,Wim Bast,T.: Analysis MDA. Bao Zhiyun Translation. People's Posts and Telecommunications Press,Beijing(2004).
2. Zhang Tian,F.:Modeling and Model Transformation of Design Pattern Based on MDA.Journal of Software 19(9), 2203-2207(2008).

3. Wang Jian,F.:Research on MVC Modeling and Model Transformation Technology Based on MDA.Chongqing University,Chongqing(2010).
4. Li Chen,F.:E-commerce order management system based on MVP architecture.Dalian Maritime University,Dalian(2017).
5. Wang Xin,F.:Design and Implementation of Universal Web Software System Development Framework Based on MVC Pattern.University of Electronic Science and Technology,Chengdu(2007).
6. Guo Jianing,F.:Design and implementation of front-end CASFront based on MVP.Tianjin University,Tianjin(2015).
7. UML Homepage,http://www.omg.org/spec/UML.last accessed 2019/5/8.
8. M. Brambilla,F.,P. Fraternali,S.:Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML.OMG,Italy(2014).
9. Marco Brambilla,F.,Andrea Mauri,S.:Eric Umuhoza.Extending the Interaction Flow Modeling Language(IFML) for Model Driven Development of Mobile Applications Front End.In:Irfan Awan,F.,
10. Muhammad Younas,S.(eds)MobiWIS 2014, LNCS, vol.8640,pp. 176-191.Springer, Barcelona(2014).