Open Journal of Applied Sciences, **2012, 2, 180-183**
doi:10.4236/ojapps.2012.23026 Published Online September 2012 (http://www.SciRP.org/journal/ojapps)

A Mapping Method of Using the Compound Use Cases to Generate User Interface

Bingbing Xia, Yue Zhang
Information Technology Department, Shandong Jiao Tong University, Ji Nan, China
Email: jennifer_xiababy@yahoo.com.cn, lv1919@163.com

Received July 7, 2012; revised August 5, 2012; accepted August 16, 2012

ABSTRACT

To support the requirement analysis of the user interfaces and the auto generation of the final user interfaces, we should improve the traditional modeling method which centered on the system and the implementation, and turn to use the modeling method which centered on the usage and the user. The compound use cases describe the system function from the user point of view. Based on the compound use cases, we can generate the final user interfaces.

Keywords: Compound Use Cases; User Interfaces; Modeling Method

1. Introduction

To support the auto generation of the user interfaces, this paper introduces the compound use cases. The compound use cases combine the UML concept of the use case model and the concept of the data flow diagram [1]. The compound use cases have its own character system; it can break down into the following parts.

1.1. Compound Use Cases Group

The compound use cases can divide into different groups, and the use case group contains one or more groups, every group has its own group name, thus the level relationships between the use cases will be more clear [2] which will help the layout of the user interface and the auto generation.

The use cases group is easy to map the menu, and the group will make the use case diagram more clear and easy to communicate with the user.

As shown in the **Figure 1**, the use case 1 and the use case 2 belong to the same use cases group: group 1 and the two use cases belong to the same menu group in the final user interface menu. The use case 3 is not belonging to group1 and the group1 will not appear in the menu.

1.2. Roles and User Group

Roles represent different and direct users. The user interfaces should meet the requirements of all the system users. Different users play different roles in the system. So the concept of roles is to differentiate different users. The roles indicate the importance of the existence of the user interface.

Different roles can divide into different user groups. The concept of user group manages the system users at different levels [3], assigns user operating privileges and divides system function at different size.

1.3. Relationships

According to the feature of the user interface design oriented, the compound use cases extend and modify the relationship between use cases of the traditional use case model, which can better support the requirement analysis and design and auto generation.

As shown in the **Figure 2**, the relationship named function use represents function use relationship, the relationship named extend represents extend relationship, the relationship named refine represents refine relationship, the relationship named navigate represents navigate relationship.

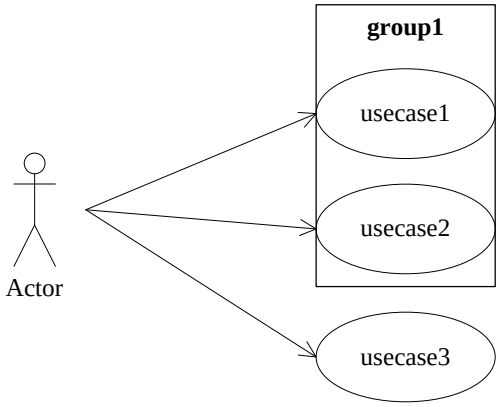

Figure 1. Use case group.

1.4. User Interface Contour Line

The compound use cases use the concept of user interface contour line, which includes the composition of the user interface in the use case diagram. The concept of the user interface happens during the requirement analysis [4]. From the user point of view, he can see the composition of the user interface during design, so he can participate in the design. From the designer point of view, this concept gives props for the representation of the navigation.

As shown in **Figure 3**, the use case navigates the user interface UI1，the rectangle represents the user interface. As shown in **Figure 4**, the user interface UI1 is composed of use case 4 and use case group group1 which composed of use case 2 and use case 3.

2. Solution

The analysis and design of the user interface using the compound use cases should follow several steps:

First, refine every compound use case, determine its function, turn the user's requirement to compound use case diagram.

Second, determine the relationships between the compound use cases and the relationships between role and the compound use case, further refine compound use cases, thus the compound use case diagram composes of role and compound use case and all kinds of relationship [5].

Third, determine the compound use case groups and user interface contour line; determine the scope of every interface. Every compound use case has its own parameters, such as the compound use case belongs to group or not, the compound use case belong to interface contour line.

The compound use case diagram can directly map into the user interface, so the concept in part I will directly map into the composition of the interface.

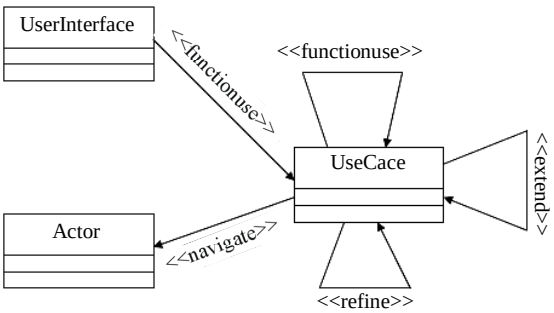

Figure 2. Relationship.

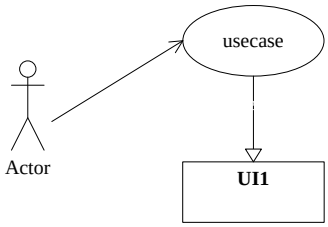

Figure 3. Navigate relationship.

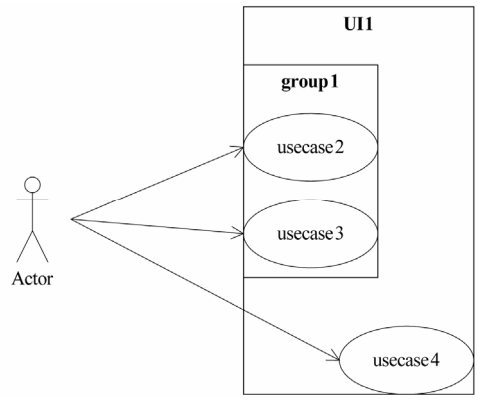

Figure 4. User interface composition.

2.1. Menu

As shown in **Figure 5**, this menu is generated from **Figure 1**. In this menu, use case 1 and use case 2 belong to the same menu group, because the two use cases belong to the same use case group. But the use case group does not appear in the menu.

2.2. Relationships

The relationships between use cases can be described as shown in **Figure 6**. Different relationship uses different arrow character system.

- Function use relationship:

This relationship is composed of the role using the compound use cases and compound use cases using compound use cases. The role using the compound use case indicates that the role has interactive relationship to the compound use case.

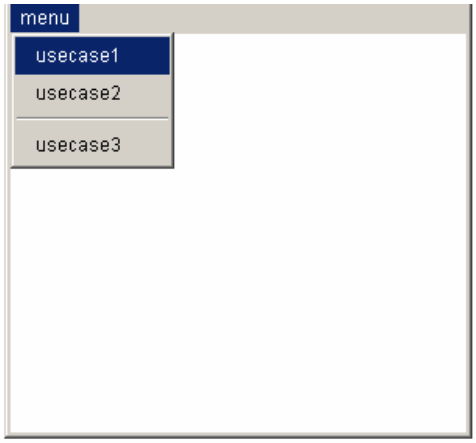

Figure 5. Menu generated from compound use case.

The function use relationship is as shown in **Figure 7**. The actor has function use relationship with use case 1 and the use case 2 has function use relationship with use case 3.

- Refine relationship:

The refine relationship represents the level relationship between the compound use cases. The use cases can divide into several sub use cases which can be divided furthermore. The use case divided is not existed indeed, it represents abstract concept [6]. The refine relationship can map into several level menus, the use case divided is not in the menu, and the sub use case is the menu item. The use cases coming from the same use case belong to the same menu group.

As shown in **Figure 8**, the use case 1 can be refined to three use cases and use case 1.2 can be refined to three use cases. In the generated menu, the use case 2.1, 2.2 and 2.3 belong to the same menu group because these use cases is refined from the same use case.

- Extend relationship:

One compound use case can extend to several sub use cases which can extend furthermore. The compound use case extended is existed. The extend relationship can map into several level menu, the use case extended is in the menu as menu item, the sub use case come from this use case will act as the sub menu.

As shown in **Figure 9**, the use case 1.2 is extended to three use cases. In the generated menu, the use case 1.2 is existed in the menu, the use case 2.1, 2.2 and 2.3 act as the sub menu of the menu named use case 1.2.

- Navigate relationship:

The navigate relationship represents the trigger and jump relationship between the user interfaces. Only the compound use case can navigate to the user interface, this is not mentioned in the traditional use case model.

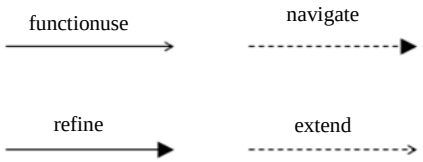

Figure 6. Relationship character system.

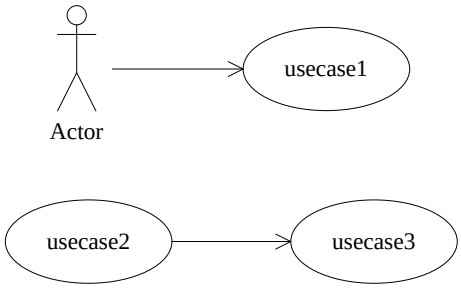

Figure 7. Function use relationship.

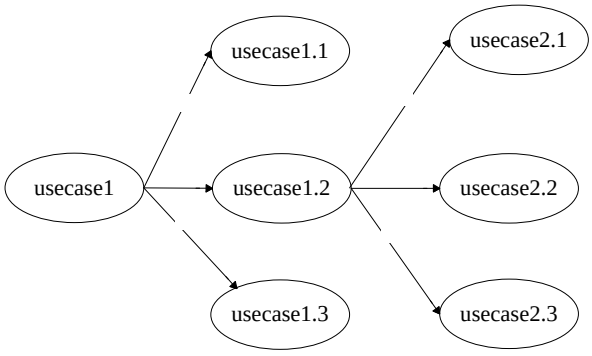

Figure 8. Refine relationship.

2.3. Compound Use Case Restrictions

The compound use case has some restrictions as follows: The roles should not have relationship to the roles. The role must have function use relationship to the compound use cases. The relationship can not exist from the compound use case to the role. The navigate relationship must exist from the compound use case to the user interface. The relationship between compound use cases must not be the navigate relationship.

The refine relationship means the compound use case refined does not exist, so if one compound use case is composed of several use cases, the relationship beginning from this use case must be the refine relationship.

The extend relationship means the compound use case extended does exist, so if one compound use case is

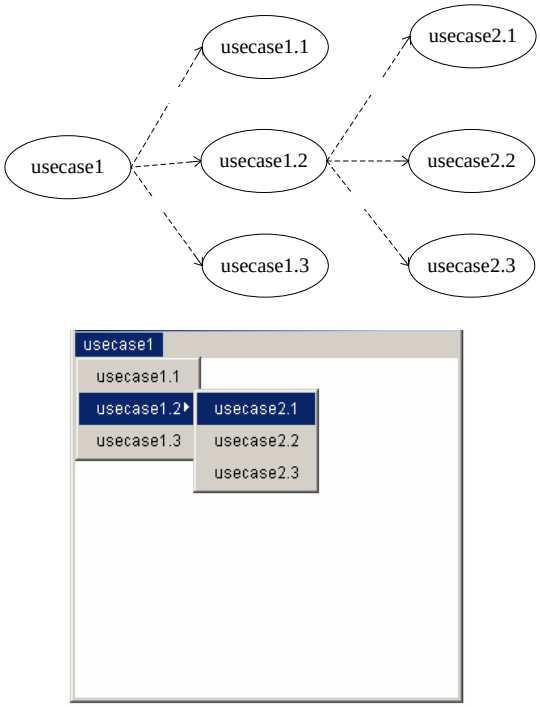

Figure 9. Extend relationship.

extended to several use cases, the relationship beginning from this use case must be the extend relationship.

If there exists non-extend or non-refine relationship between compound use cases, the next relationship from the beginning use case of these relationships must not be extend or refine relationship.

3. Conclusion

The concept of compound use case combines the use case diagram in the UML object-oriented analysis and design method, uses in the requirement analysis period based on the UML object-oriented software engineering method [7], describes the user requirement of user interface. The compound use case is a character system and modeling tool. Using the compound use cases, we can support the whole analysis and design process from user interface requirement to the final user interface.

4. Acknowledgements

I would like to express my deepest gratitude to TianRui, who helped me a lot to complete this paper. Second, I will extend my heartfelt gratitude to teacher Youping Dong who helped me a lot during my work.

REFERENCES

[1] G. Raghava, "Designing User Interfaces from Use Cases and Modeling Interface Behaviors with Statecharts [DB/OL]". http://www.umlchina.com

[2] S. Egbert, "Model-Based User Interface Software Tools: Current State of Declarative Models [R]," GVU Tech Report, Graphics, Visualization and Usability Center, Georgia Institute of Technology.

[3] N. J. Nunes and J. F. E. Cunha, "Wisdom—A UML Based Architecture for Interactive Systems [C]," *Proceedings of DSV_IS'*2000, *Limerick, Lecture Notes in Computer Science* (*LNCS*), Springer-Verlag, New York, 2000.

[4] N. J. Nunes and J. F. E. Cunha, "Towards a UML Profile for Interaction Design: The Wisdom Approach [C]. UML 2000—The Unified Modeling Language. Advancing the Standard," *Proceedings of* 3*rd International Conference*, York, October 2000, pp. 101-116.

[5] P. P. da Silva and N. W. Paton, "UMLi: The Unified Modeling Language for Interactive Applications [C]. The Unified Modeling Language: Advancing the Standard, 3*rd International Conference on the Unified Modeling Language*, York, 2-6 October 2000," In: A. Evans, S. Kent and B. Selic, Eds., *Lecture Notes in Computer Science* (*LNCS*), Vol. 1939, Springer, Berlin, 2000, pp. 117-132.

[6] M. Elkoutbi and R. K. Keller, "User Interface Prototyping Based on UML Scenarios and High Level Petri Net [R]," *Lecture Notes in Computer Science*, Vol. 1825, 2000, pp. 166-184.

[7] J. Vanderdonckt, "Using Data Flow Diagrams for Supporting Task Models[C]," *Companion Proceedings of* 5*th Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS*'98, Abingdon, 3-5 June 1998.

