Aspect-Oriented Extension for Capturing Requirements in Use-Case Model

Chanwit Kaewkasi[1], Wanchai Rivepiboon[1]

[1] Software Engineering Laboratory, Chulalokorn University,
254 Phyathai Road, Patumwan, Bangkok 10330, Thailand
chanwit@customix.net, wahchai.r@chula.ac.th

Abstract. Early Aspects is a concept that applies an aspect-oriented (AO) paradigm to the requirements engineering. Aspect-Oriented Requirements Engineering (AORE) can be considered as an important role in the early phase of aspect-oriented software development (AOSD). Crosscutting concerns provide modularized concept for tangled representation of the software. There are several works in the AOSD area that emphasized on the design and implementation level. In this paper, we develop novel techniques for using AO concept in the early phase of the use-case driven software development process. Our approach employs an AO concept to capture both functional, and nonfunctional requirements. Several notations are introduced to extend the use-case model of the UML.

1 Introduction

Separation of concerns is a research topic that has been raised in the last few years. Aspect-oriented paradigm is for handling crosscutting concern throughout the software development life cycle. This work is basically related to aspect-oriented requirements engineering (AORE). AORE is an early phase of the software development process in the aspect-oriented software development process. AORE is intended to handle tangled representation of the software artifacts at the requirements level.

In the use-case driven software development process [4], which is service-oriented, functional requirements can be modeled as use cases of the system-to-be. A use case can be simply considered as a service. A use case is a sequence of activities that are performed by a set of objects in the system. Thus, objects work together to serve the service to their stakeholder. To better reuse services for the new software project, the customizable services are needed. Generally, we specify properties of the system as a set of nonfunctional requirements (NFRs). Handling of crosscutting concerns helps managing the system changes. The system that is separated this concern effectively can be improved understandability and maintainability through out the cycle of development.

The use-case driven approach provides several benefits with the functional requirements capturing, such as hiding the system's complexities, representing the functional requirements as a set of services that provides to their actors, etc. [4]. Our approach modifies the metamodel of the UML. We mainly add a set of new notations to the use-case package of the UML.

This work moves toward the AOSD as an early-aspect software engineering to deal with the requirements phase of the process. Our work aims at improving the process for capturing and representation both functional, and NFRs. This work improves, and refines their concept of AORE in the use-case model. Our work shows that it is possible to model requirements in the AO approach using our new notations, and can be combined with the de facto standard software modeling language, the UML.

The rest of this paper is organized as the following: section 2 discuss about motivation and related works to out approach. Section 3 presents our new notations, discusses how the use-case model of the UML is modified. Section 4 concludes and discusses further works.

2 Motivation

Research in the early phase activity of the software development with AO paradigm has been increasing. AOSD is a technology based on the concept of aspect-oriented programming (AOP) [5], and the multi-dimensional separation of concerns [3]. There are several works involved AOSD in many phases of software development. A number of AOP languages have been proposed [9].

AORE of component-based software was proposed by Grundy [2]. His work characterizes different aspects of the system that each component provides to the user and its related components. Recent works [1], [6], [8] focused on the early phase of software development with AO paradigm. The early aspect concept presented in [8] is a general AO model based on viewpoint-oriented requirements engineering. The model supports separation of crosscutting properties for identification their mapping and influence in the later phases of software development.

Crosscutting quality attributes handling at the requirements phase of the software development using UML was proposed in [1]. The work presented weaving use cases using the traditional notations of the use-case model. The revised set of notation of [1] was proposed in [6]. The authors introduced a number of stereotypes for using in the use-case model of the UML.

We have found that the extension to AORE proposed by [1], [6] does not adequate to handle some kind of aspects. The authors proposed a set of UML stereotypes to help requirements engineers capturing both functional and nonfunctional requirements, but their approach did not concern on the simplicity and flexibility to transform those software artifacts to use in the next phase of the software development. In this paper, modification to the metamodel of the UML is presented. This approach is to make support of AO paradigm for the requirements model.

3 Extension to the Use-Case Package

Our approach combines the aspect-oriented paradigm to the use-case model. We add several extensions to the use-case package in the metamodel of the UML [7]. The use-case package is a subpackage of the behavioral package of the metamodel. The key elements are the use case and actor notations. To extend its functionality for capturing NFRs with the concept of the AO, an advice case, a use-case selector, and a pointcut association, are introduced here.

3.1 Advice Case

An advice case is defined as a specialization of the classifier from the UML metaclass. It defines a sequence of actions, and has similar characteristics to a use case; except that it cannot be performed directly by the actor. A concept of an advice case follows the concept of *advice* in the AOP [5]. An advice case can be modeled in with a notation of use case and <<advice case>>. It can also be modeled using a vertical-half-ellipse shape. Graphical representations of an advice case are displayed in figure 2.

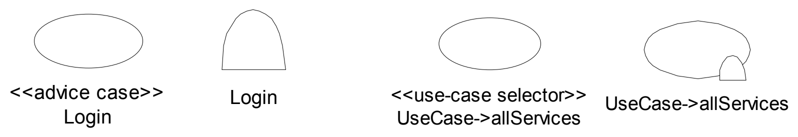

Fig. 1. Notations of an advice case and a use-case selector.

3.2 Use-case Selector

In order to perform the associated advice case, the system should know *when* to perform the advice case. According to AOP, a pointcut is a set of selected join points of the system [5]. A pointcut defines *what* will be crosscut, and when. In our approach, we use a use-case selector to define what to be crosscut. A use-case selector contains an OCL expression, and uses it to evaluate which use-case to be selected.

A use-case selector can be displayed as a use-case notation with attached <<use-cases selector>> stereotype. It is also represented as a use-case with a little vertical-half-ellipse attaching at the right corner of it. Figure 2 shows both representations of graphical notation of a use-cases selector.

3.3 Pointcut Association

A pointcut association links its related use-case selector to an advice case. This kind of association must be labeled with a stereotype to indicate *when* the system should perform appropriate advice case. Figure 4 shows the use of the *entering* pointcut association incorporating with the use-case selector and the advice case *Login*.

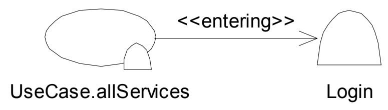

Fig. 2. This shows a working combination of a use-case selector, a pointcut association, and an advice case. The pointcut association labeled with <<entering>> forces the system to perform the advice case "Login" *before* performance of all use cases in the current model.

4 Conclusion and Future Works

We have presented in this paper an approach to integrate the AO paradigm with the use-case model for capturing requirements in the early phase of software development. Our approach models requirements into a set of aspects that crosscut sequence of use cases. We have introduced three AO notations to the use-case model of the UML, advice case, a set of pre-defined pointcut associations, and a use-case selector. The methodology to make this AO use-case model into the analysis and design phase, defining more pointcut associations, defining the well-formness definition for our notations are to be done. We hopefully make more seamless integration of our approach through all phases of the unified process.

References

1. Araujo, J., et al. *Aspect-Oriented Requirements with UML*. in *UML 2002*. 2002.
2. Grundy, J. *Aspect-oriented Requirements Engineering for Component-based Software Systems*. in *4th IEEE International Synposium on Requirements Engineering*. 1999. Limerick, Ireland: IEEE Computer Society Press.
3. IBM, *MDSOC: Software Engineering using HyperSpaces. http://www.research.ibm.com/hyperspace/*, IBM Research.
4. Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. The Addison-Wesley Object Technology Series. 1999: Addison-Wesley.
5. Kiczales, G., et al. *Aspect-Oriented Programming*. in *Proceedings European Conference on Object-Oriented Programming*. 1997: Springer-Verlag.
6. Moreira, A., J. Araujo, and I. Brito. *Crosscutting Quality Attributes for Requirements Engineering*. in *14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002)*. 2002. Italy: ACM Press.
7. OMG, *The Unified Modeling Language Specification version 1.4*. 2001, Object Management Group. http://www.omg.org/uml.
8. Rashid, A., et al. *Early Aspects: a Model for Aspect-Oriented Requirements Engineering*. in *IEEE Joint Conference on Requirements Engineering*. 2002. Essen, Germany: IEEE Computer Society Press.
9. Xerox, *Aspect/J Homepage. http://www.aspectj.org/*, Xerox Parc.