Representing Aspects in Design

Saqib Iqbal, Gary Allen
Department of Computing and Engineering
Univeristy of Huddersfield, Huddersfield, HD1 3DH, UK
s.iqbal@hud.ac.uk, g.allen@hud.ac.uk

Abstract

Identification of cross-cutting concerns (Aspects) in the earliest phases of software development has gained in popularity over recent years. Many approaches have been suggested for identifying and representing Aspects in abstraction and design structures. Since these approaches are still relatively immature, shortcomings such as overlooking or not properly locating Aspects have been noted in almost all of these approaches. This research proposes an Aspect-Oriented Software Development Model which helps identifying Aspects at the early stages of software development and guides the way to interpret these Aspects into Design elements and finally provides a support to weave Aspects into Base program.

Keywords: Aspect-Oriented Programming, AOSD Modeling, Identification of Aspects, Representation of Aspects

1. Introduction

Many essential system requirements, such as efficiency, security, fault-tolerance, and synchronization of threads, are difficult to handle, especially in the implementation of large-scale systems. The slightest mishandling of any of these requirements can result in a big problem, or even disaster in the case of safety critical systems. Such requirements are rarely limited to one module or unit of a system, rather their implementation spreads over a set of modules or sub-modules. Their implementation, therefore, also involves more than one programming unit. Hence their code is scattered and tangled across the whole system, and that is why they are known as cross-cutting (or Aspectual) concerns of the system. The complex, yet important nature of these Aspects has forced software engineers to address them separately from the base program. Aspect-oriented programming (AOP) [2] has been proposed as a programming paradigm to handle these cross-cutting concerns. Since its inception, a lot of work has been carried out on the better implementation of Aspects. There are plenty of tools and technologies like AspectJ, AspectWorkz, and Spring available which implement Aspects differently according to the requirements and environments.

Aspects are usually handled in the implementation phase. Their identification usually relies on the strength of domain knowledge of the implementer. There has been very little work in identifying Aspects at the earlier stages of software development, although some techniques have been proposed, like the AORE (Aspect-Oriented Requirements Engineering) model by Rashid et al [1] which builds on View Point Model [1], and the COSMOS [3] model by S. Sutton which proposes a technique to capture concerns in the early stages. The problem with both of these techniques is that they do not specifically capture cross-cutting concerns; rather they talk about general concerns of the system. Some of other related work can be found in [4], [6] and [8]. All these techniques and models either address modeling of Aspects or identifying general concerns including non cross-cutting concerns (non Aspects). There are some approaches which represent Aspects in Use Case models. For example, Saiki and Keya propose generating a use case model for Non functional requirement (NFR or Aspects) [5]. Araújo and Coutinho proposed developing a vision document based on a viewpoint-oriented method to separate Aspects from basic concerns [7], and Georgia et al have proposed extensions in UML showing use cases in the use case model and suggested techniques to implement Aspects as use cases [9]. All these proposed approaches lack a complete model of representing Aspects from system initialization to its implementation.

Our research focuses on developing efficient procedures and well-defined set of activities to identify, represent and weave Aspects in the Software Design. We have proposed an Aspect-Oriented Software Development model (Figure1) that represents Aspect from the initialization of software to its implementation. It suggests the identification of Aspects in the Use Case Model and Sequence Diagrams of the system. Use cases which involve multiple use cases like included or extended use cases may be considered as candidate aspects since they have the probability of crosscutting representation in design as well as in implementation. Similarly, the objects which have communication with multiple objects and which are represented in multiple sequence diagrams may also be regarded as candidate aspects. Proper specification of the candidate aspects can help identifying actual Aspects. Once we have identified Aspects which are to be implemented in the system, we can include them in class diagram to represent their relationship with classes of Base program.

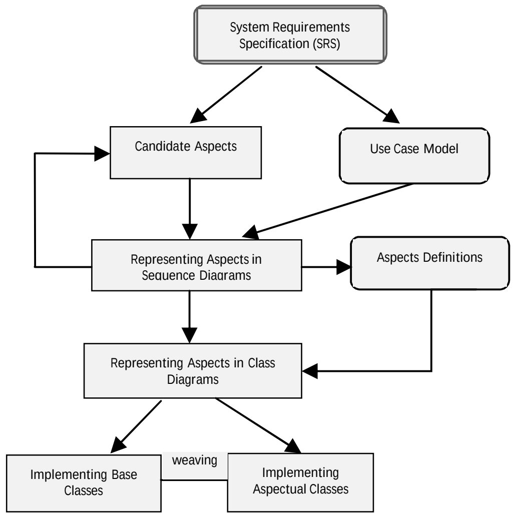
System Requirements Specification (SRS)
Candidate Aspects
Use Case Model
Representing Aspects in Sequence Diagrams
Aspects Definitions
Representing Aspects in Class Diagrams
Implementing Base Classes
weaving
Implementing Aspectual Classes

Figure 1: AOSD Model

The proposed AOSD model represents Aspect in all phases of software development and can help identifying and tackling all the system Aspects in traceable and evolvable manner.

2. References

[1] A. Rashid, A. Moreira, and J. Araujo, "Modularisation and Composition of Aspectual Requirements," presented at 2nd International Conference on Aspect Oriented Software Development (AOSD), Boston, USA, 2003.

[2] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. XEROX PARC Technical Report, SPL97-008 P9710042. February 1997.

[3] S. M. Sutton, "Concerns in a Requirements Model - A Small Case Study," presented at Early Aspects 2003 Workshop: Aspect-Oriented Requirements Engineering and Architecture Design (held with AOSD 2003), Boston, USA, 2003.

[4] M. Katera and S. Katz. Architectural views of Aspects. In Proceedings of the International Conference on Aspectoriented Software Development, pages 1–10, 2003.

[5] M. Saeki and H. Kaiya. Transformation Based Approach for Weaving Use Case Models in Aspect-Oriented Requirements Analysis. In The 4th AOSD Modeling With UML Workshop, Oct. 2003.

[6] X. Wang and Y. Lesperance. Agent-oriented requirements engineering using congolog and i*. In Submitted to AOIS-2001, Bi-Conference Workshop at Agents 2001 and CAiSE'01., 2001.

[7] Araujo, J., Coutinho, P.: Identifying Aspectual use cases using a viewpoint-oriented requirements method. In: Early Aspects 2003: Aspect Oriented Requirements Engineering and Architecture Design, Workshop of the 2nd Intl. Conference on Aspect-Oriented Software Development, Boston, MA (2003)

[8] J. Castro, M. Kolp, and J. Mylopoulos, ‘‘Towards requirements-driven information systems engineering: the TROPOS project,’’ Information Systems, vol. 27, pp. 365–389, 2002.

[9] A. Rashid, B. Tekinerdo˘gan, A. Moreira, J. Ara´ujo, J. Gray, J. G. Wijnstra, and P. Clements. Early Aspects: Aspectoriented requirements engineering and architecture design. In Workshop at AOSD-2002, 2002.