Chapter 8
An Algebraic Approach for the Specification and the Verification of Aspect-Oriented Systems

Arsène Sabas
Université de Montréal, Canada

Subash Shankar
City University of New York (CUNY), USA

Virginie Wiels
ONERA – The French Aerospace Lab, France

John-Jules Ch. Meyer
Universiteit Utrecht, The Netherlands

Michel Boyer
Université de Montréal, Canada

ABSTRACT

Aspect-Oriented (AO) Technology is a post-object-oriented technology used to overcome limitations of Object-Oriented (OO) Technology, such as the cross-cutting concern problem. Aspect-Oriented Programming (AOP) also offers modularity and traceability benefits. Yet, reasoning, specification, and verification of AO systems present unique challenges, especially as such systems evolve over time. Consequently, formal modular reasoning of such systems is highly attractive as it enables tractable evolution, otherwise necessitating that the entire system be re-examined each time a component is changed or is added. The aspect interactions problem is also an open issue in the AOP area. To deal with this problem, the authors choose to use Category Theory (CT) and Algebraic Specification (AS) techniques. In this chapter, the authors present an aspect-oriented specification and verification approach. The approach is expressive and allows for formal modular reasoning.

DOI: 10.4018/978-1-4666-6026-7.ch008

1 INTRODUCTION

Aspect-Oriented Programming has emerged in recent years as a new paradigm that provides a set of concepts that allow programmers to modularize their applications in order to provide better Separation of Concerns (SoC). Thanks to AOP, programmers can implement different concerns in well-defined entities called aspects by breaking the inherent dependencies that can exist between the different program modules. By using AOP, programmers thus increase the maintainability and the readability of their programs. AOP, giving rise to programming languages such as AspectJ Kiczales et al. (1997), evolved from a programming activity to a full-blown software engineering process, having the goals of preserving modularity and traceability, which are two important properties of high-quality software.

Yet, there are also many challenges in AO Technology. Reasoning, specification, and verification of AO programs present unique challenges especially as such programs evolve over time. Consequently, modular reasoning of such programs is highly attractive as it enables tractable evolution, which would otherwise require that the entire program be re-examined each time a component is changed or is added. It is well known in the literature, however, that modular reasoning about AO programs is difficult due to the fact that the applied aspects often alter the behavior of the base components Khatchadourian et al. (2008). The same modular reasoning difficulties are also present in the specification and verification phases of software development processes. To the best of our knowledge, AO modular specification and verification is a poorly studied subject and constitutes an interesting open research field.

Aspect interaction is also a major concern in the aspect-oriented community. Detection and resolution of undesirable aspect interactions is an important open research field and we believe that formal models are needed to handle unexpected interactions. Most AO verification approaches are based on a strategy of detection and correction. Although these detection approaches are relevant for AO software reliability, we believe that they are time and cost consuming. It is good to detect and correct system failures, but it is better to first prevent them; consequently, we advocate a prevention policy to be integrated at the specification phase. We believe indeed that this will make the verification phase timeless and costless.

To help reason about AO systems, the use of formal methods becomes desirable. Formal methods are essential to support quality, modifiability and reusability by formal concepts for data abstraction and modularity Ehrig et al. (1992). We believe that CT and AS can help us achieve these goals. CT is a good and powerful tool for the modularization of system components which can be considered as objects of a category. It introduces the notion of morphism, which can be used as a means to study and to implement interactions within these components. Moreover, it has construction operators that allow to structurally compose these components to form the complete system. To deal with the resolution of undesirable interactions between aspects and base components, we decided to rely on CT and AS to model and verify AO systems. By using CT, we can take advantage of the structure of a system specification to carry out the verification task and to infer some desirable properties on the global system. The principle of this modular verification is as follows: for a morphism m: $\mathrm{MOD}_1 \rightarrow \mathrm{MOD}_2$, if a property P is true in MOD_1, then $m(P)$ is true in MOD_2. For example, suppose we want to prove a property (by model-checking or a theorem proving technique) on the module MOD_S representing the entire system and we have the morphisms $\mathrm{MOD}_3 \rightarrow \mathrm{MOD}_S$ and $\mathrm{MOD}_4 \rightarrow \mathrm{MOD}_S$. This property may result from the conjunction of two lemmas; each lemma may be proved in the smallest modules (MOD_3 and MOD_4) and then translated to MOD_S.

The main contributions of this chapter are the following: