Chapter 8
Software Evolution, MDA and Design Pattern Components

INTRODUCTION

The success of MDA depends on the definition of model transformations and component libraries which make a significant impact on tools that provide support for MDA. MDA is a young approach and several technical issues are not adequately addressed. For instance, existing MDA-based CASE tools do not provide adequate support to deal with component-based reuse (CASE, 2009). In light of this, we propose a metamodeling technique to reach a high level of reusability and adaptability of components.

Reusability is the ability to use software elements for constructing many different applications. An ideal software reusability technology should facilitate a consistent system implementation, starting from the adaptation and integration of “implementation pieces” that exist in reusable components library. Software reusability has two main purposes: to increase the reliability of software and to reduce the cost of software development. Most current approaches to object oriented reusability are based on empirical methods. However the most effective forms of reuse are generally found at more abstract levels of design (Krueger, 1992).

In MDA, software reusability is difficult because it requires taking many different requirements into account, some of which are abstract and conceptual, while others, such as efficiency are concrete. A good approach for MDA reusability must reconcile models at different abstraction levels.

In this chapter, we analyze how to define reusable components in a way that fits with MDA and propose a megamodel for defining MDA components. Considering the relevant role that design patterns take in software evolution we exemplify MDA components for them.

We propose a megamodel to define families of design pattern components by means of PIM-, PSM- and ISM-metamodels and their interrelations. Instances of the megamodel are reusable components that describe specific design patterns at different levels of abstraction (PIMs, PSMs and ISMs). They can be

DOI: 10.4018/978-1-61520-649-0.ch008
viewed as megacomponents that allow defining in a concise way as many components as different pattern solutions can appear. We analyze metamodel transformations of both PIMs into PSMs, and PSMs into ISMs (Favre, & Martinez, 2006).

The traditional techniques for verification and validation are still essential to achieve software quality. We describe foundations for constructing formalizations of design pattern component. We define a megamodel based on MOF-metamodels and metamodel-based model transformations and show how to formalize them by using the metamodeling notation NEREUS. This notation, as we had said, can be viewed as an intermediate notation open to many other formal languages (Favre, 2006) (Favre, 2005). We illustrate our MDA-based approach by using the Observer design pattern.

Considering design patterns is a relevant technique in software development, in particular in forward and reverse engineering processes we include some references to related work and remark the contribution of an MDA approach.

**RELATED WORK**

This section shows the evolution of design pattern techniques and remarks the advantages of an MDA approach to define design pattern components.

In (Budinsky, Finni, Vlissides, & Yu, 1996) a tool to automatically generate code of design patterns from a small amount of information given by the user is described. This approach has two widespread problems. The user should understand “what to cut” and “where to paste” and both cannot be obvious. Once the user has incorporated pattern code in his application, any change that implies to generate the code again will force it to reinstate the pattern code in the application. The user cannot see changes in the generated code through the tool.

Florijn, Meijers, and van Winsen, (1997) describe a tool prototype that supports design pattern during the development or maintenance of object-oriented programs.

Albin-Amiot and Guéhéneuc (2001) describe how a metamodel can be used to obtain a representation of design patterns and how this representation allows both automatic generation and detection of design patterns. The contribution of this proposal is the definition of design patterns as entities of modeling of first class. The main limitation of this approach concerns the integration of the generated code with the user code.

Judson, Carver and France (2003) describe an approach to rigorous modeling of pattern-based transformations that involve specializations of the UML metamodel to characterize source and target models.

Kim, France, Ghosh, and Song (2003a) describe a metamodeling approach to specify design patterns using roles. They analyze the characteristics of object-based roles and generalize them. Based on the generalized notion of a role, they define a new notion of a model role which is played by a model element. The approach is intended to be easy to use and practical for the development of tools that incorporate patterns into UML models.

Kim, France, Ghosh, and Song (2003b) describe a metamodeling approach that uses a pattern specification language called Role-Based Modeling Language (RBML). A pattern specification defines a family of UML models in terms of roles, where a role is associated with a UML metaclass as its base. RBML uses visual notations based on the UML and textual constraints expressed in OCL to specify patterns properties. The RBML allows specifying various perspectives of design patterns such as static structure, interactions and state-based behavior.