Chapter IV

A Framework Based on Design Patterns: Implementing UML Association, Aggregation and Composition Relationships in the Context of Model-Driven Code Generation

Manoli Albert, Universidad Politécnica de Valencia, Spain

Marta Ruiz, Universidad Politécnica de Valencia, Spain

Javier Muñoz, Universidad Politécnica de Valencia, Spain

Vincente Pelechano, Universidad Politécnica de Valencia, Spain

Abstract

This chapter proposes a framework based on design patterns to implement UML association, aggregation, and composition relationships. To build the framework, we propose a semantic interpretation of these concepts that avoids the ambiguities introduced by UML. This interpretation is achieved.
by using a set of properties that allows us to characterize these kinds of relationships. Once the semantics of the relationships have been defined, we propose a framework based on design patterns for the systematic generation of a software representation. The framework is based on the properties that characterize the relationships. It provides a high-quality solution and introduces important benefits with regard to other existing implementation approaches. This work proposes an implementation strategy that defines a set of mappings between the conceptual abstractions and the proposed framework. This strategy enables the automatic instantiation of the framework. Finally, to validate the proposal, we present a C# implementation of a collaboration pattern. Collaboration patterns are analysis patterns constituted by two classes that are related by an association, an aggregation or a composition relationship.

**Introduction**

Current development methods and tools are focused on model-driven software development (MDSD) processes. In particular, the model-driven architecture (MDA) proposal of the object management group (OMG) constitutes an approach for the development of software systems that is based on a clear separation between the specification of the essential system functionalities and the implementation of this specification through the use of specific implementation platforms. MDA tries to raise the abstraction level in software development by giving more relevance to conceptual modeling. Models with a high level of abstraction (platform independent models, PIM) are translated into models that are expressed in terms of specific implementation technologies (platform specific models, PSM). The PSM can be used to generate automatically the application code. A practical application of MDA requires more mature techniques and tools than those that are currently available. In order to achieve the MDA goals, we consider that it is necessary to provide:

- **Conceptual modeling abstractions with a precise semantics:** The precision of the modeling abstractions is a key characteristic in building appropriate PIM. Rich conceptual models are needed to define transformations that guarantee that the generated code is functionally equivalent to the specification.

- **Implementation techniques:** Frameworks and design patterns facilitate the implementation of the modeling abstractions in target implementation languages, the definition of transformations, and the production of high-quality solutions.