Chapter 12
A Test-Driven Approach for Metamodel Development

A. Cicchetti
Malardalen University, Sweden

D. Di Ruscio
University of L'Aquila, Italy

A. Pierantonio
University of L'Aquila, Italy

D.S. Kolovos
The University of York, UK

ABSTRACT

Model Driven Engineering (MDE) is increasingly gaining acceptance in the development of complex systems as a mean to leverage the abstraction level and render business logic resilient to technological changes. Metamodels precisely define the constructs and underlying well-formedness rules for modeling languages. However, the definition of a metamodel is intrinsically complex since it must be precisely tailored according to the specific purpose the models must have.

The paper proposes a test-driven development process for metamodels, where models are used a) to validate metamodel expressiveness in the early stages of their definition, and b) to convey feedback to designers as a guidance for further refinement and evolution.

INTRODUCTION

Model Driven Engineering (MDE) (Schmidt, 2006) is increasingly gaining acceptance as a mean to leverage abstraction and render business logic resilient to technological changes. Coordinated collections of models and modelling languages are used to describe software systems on different abstraction layers and from different perspectives. In general, domains are analysed and engineered by means of *metamodels*, i.e., coherent sets of interrelated concepts. A model is said to *conform* to a metamodel, or in other words it is expressed by the concepts encoded in the metamodel, and

DOI: 10.4018/978-1-61350-438-3.ch012

model transformations occur when models are translated into other artifacts such as other models, code and documentation.

Metamodels are vital entities for designers and tool implementors as they define useful standards that enable tools and models to work together portably and effectively. Since almost any artifact involved in a model-driven development process is depending on the considered metamodels (Kurtev, Berg, & Jouault, 2006), defining a metamodel as a whole with little or no feedback is practically unrealistic. In fact, the expressiveness of the metamodel, i.e., the amount of detail which has to be captured for each concept, depends predominately on the kind of applications (e.g., model-to-model and model-to-code transformations) the designer is aiming at.

Little guidance exists on creating modeling languages (Kelly & Pohjonen, 2009). Designing and implementing a metamodel in a consistent manner requires an in-depth understanding of the problem domain and enough solution domain expertise to foresee the necessary information that instance models will need to capture. In practice, a metamodel often evolves towards a final form only after it undergoes an iterative restructuring and refinement process. Each iteration consists of extending and refining the set of available features and adapting the corresponding model transformations and tools which are tightly coupled with the metamodel. Possible shortcomings, such as the inability to correctly generate an artifact fragment, must be conveyed to the designer in order to enhance the metamodel. Moreover, such a step-wise process also continues after the first delivery, based on user feedback.

Test-driven development (Beck, 2000) (TDD) is an increasingly popular approach for building systems with reliability, understandability, and maintainability requirements. In this paper, we present an iterative bottom-up approach to metamodel development based on TDD where models are considered as test cases that are validated against incremental metamodel implementations. In fact, a model can be seen as an unambiguous requirement that the metamodel and its implementation must satisfy: *a)* once the current version of the metamodel successfully captures the model *b)* the designer can then take a step back, survey the landscape that was created *c)* refactor and then move on to the next requirement. As such, the metamodel development process consists of small incremental steps which provide crucial feedback. These small steps eventually lead to the definitive metamodel with a better and more verifiable formalization. In contrast with similar approaches (Paige, Brooke, & Ostroff, 2004), which mainly focus on formally checking terminal models against a metamodel for conformance, the proposed technique aims at consistently defining a metamodel at a level of abstraction which is the most appropriate for the envisioned goals. The approach has been validated by defining and implementing the beContent metamodel, a full-scale domain-specific modeling language for data-intensive Web applications (Cicchetti, Di Ruscio, Eramo, Maccarrone, & Pierantonio, 2009 ; Cicchetti, Di Ruscio, Iovino, & Pierantonio, 2011).

The remainder of the paper is structured as follows. The motivation for this work is discussed in the next section. Section 3 presents the TDD-based approach, and an agile framework for metamodel development alongside the necessary underlying infrastructure. Then, in Section 4 a real-world application of the process is discussed. Finally, the work is compared with related research in the field of language engineering and development and some conclusions are drawn.

MOTIVATING SCENARIO

In the MDE vision (Bézivin, 2005) abstractions are provided by models, which are intended to be approximations of reality: each application is described by means of domain-specific models which are refined step-by-step to obtain the implementation. The refinement steps are automated by