Beyond Stereotyping: Metamodeling Approaches for the UML

Ansgar Schleicher, Bernhard Westfechtel
Department of Computer Science III
Aachen University of Technology
Ahornstr. 55, 52074 Aachen
(schleich|bernhard)@i3.informatik.rwth-aachen.de

Abstract

The UML is currently being used as the universal technique for modeling object-oriented applications across a wide range of domains. Developing a truly adequate uniform modeling technique in the face of these diverse domains seems an unsolvable quest and contrasts domain specific software engineering activities. Recently, many adaptations to the UML have been made to reflect a domain’s world view. These adaptations often exceed the UML’s own extension mechanisms and result in yet another urban UML slang.

However, domain-specifically adapting the UML metamodel becomes increasingly important in the context of model checking and code generation mechanisms. Therefore solutions should be found to fully support metamodeling within the UML and UML CASE tools.

The paper discusses and evaluates the UML’s inherent as well as proprietary metamodeling approaches and will provide domain driven ideas for a meta-modeling approach for a diversely used Unified Modeling Language

1. Introduction

After a wide variety of object-oriented modeling languages was created particularly in the 90’s, the UML [3] was introduced as a standard notation in order to overcome the upcoming confusion. To make it a general-purpose modeling language usable in a rich spectrum of application domains, the designers of the UML decided to include a comprehensive set of modeling techniques for analysis and design as well as structural and behavioral modeling. In this way, they hoped to offer UML users all support they require for their specific applications.

However, it was recognized early that it is difficult to develop a single modeling language suiting the needs of different application domains. This seems to contrast with the goals of domain-specific software engineering which is concerned with the design of modeling languages that adequately support the concepts of a specific domain.

As a compromise between the requirements for a standard notation and for domain-specific modeling, the UML was designed as an extendible modeling language. In this way, the users of the UML would be able to tailor the language to their specific requirements by introducing domain-specific model elements. On the other hand, these extensions would be performed in a way that conforms with the UML standard.

In this paper, we compare different approaches to extending the UML. We are interested in how the UML may be extended such that

- the extensions are easy to understand (readability),
- the semantics of domain-specific concepts may be expressed (expressive power),
- the extensions may be made restrictive (restrictive power),
- domain-specific constraints may be easily checked (checkability), and
- the extensions still conform to the UML, i.e., they must not redefine UML model elements in arbitrary ways or define completely new UML elements (conformance).

Clearly, the requirements to extension mechanisms depend on the respective application. In this paper, we study applications that require domain-specific models with well-defined semantics. This is crucial when models are required to be executable or code has to be generated from a model. Semantic domain modeling puts high demands particularly on expressive and restrictive power as well as on checkability. In addition, readability and conformance are general requirements that have to be addressed anyway.

The rest of this paper is structured as follows: In Section 2, we introduce a case study (workflow modeling) that is used in Section 3 as a running example to present and compare different approaches to extending the UML.