Enforcing Modeling Guidelines in an ORDBMS-based UML-Repository

N. Ritter and H.-P. Steiert
University of Kaiserslautern, Germany

Due to its rich set of modeling concepts and its broad application spectrum the Unified Modeling Language (UML) has become widely accepted for modeling many aspects of software systems. Since UML is not related to any particular design method, each software development project has to establish its own modeling guidelines. Hence, tool support is needed for guiding the developer throughout the modeling process and for enforcing project-related integrity of UML models. In this chapter, we present our approach for enforcing guidelines in UML-based software development processes. For managing UML models, we implemented a UML repository on top of an object-relational database management system (ORDBMS). Guidelines are expressed as OCL constraints and are enforced either automatically, i.e., by the UML repository, or on user demand. For this purpose, we take advantage of ORDBMS query facilities for checking guidelines by automated mapping of OCL constraints to SQL expressions.

INTRODUCTION

In our SENSOR project (Subproject A3 Supporting Software Engineering Processes by Object-Relational Database Technology of the Sonderforschungsbereich 501 Development of Large Systems by Generic Methods funded by the German Science Foundation) we consider new object-relational database technology for software systems which have to support data man-
management tasks in software engineering projects. In detail we aim at two goals. First, we are developing a shared UML repository (UML, Unified Modeling Language, (OMG, 1997a, 1997b; UML Specification)) based on an object-relational database management system (ORDBMS) (Stonebraker & Brown, 1999) in order to support cooperation of developers and reuse of design. Second, we want to generate database schemas and object-oriented database application programming interfaces (API) for engineering applications from a graphically specified UML model. This paper deals with some important aspects of our UML repository: exploiting OCL constraints for preserving consistency of UML models and for enforcing guidelines during the modeling process.

UML is becoming a de-facto standard for object-oriented modeling. The Object Management Group (OMG) has adopted the UML into its Object Management Architecture (OMA). Furthermore, UML has become broadly supported by the vendors of graphical modeling tools. In comparison to other information models, e.g., the Entity-Relationship-model (Chen, 1976), UML has lots of advantages. It is object-oriented and object-orientation has become the leading software development technology. Also, UML offers a large set of structural modeling elements including class structures and several options to define the semantics of relationships. In addition, it comes along with modeling elements for describing the behaviour of a system and for state-oriented aspects. The OCL (Object Constraint Language, (OMG, 1997c; Warner & Kleppe, 1999)) enables developers to specify constraints in a descriptive manner. Unfortunately, OCL is only weakly represented in many descriptions of UML (Linington, 1999). In this chapter, we will focus on the use of OCL for our purposes.

Our implementation of a UML repository is based on the UML meta-model and is implemented by exploiting an ORDBMS. The enhanced type system, the powerful SQL facilities and the extensibility features of ORDBMSs have proven to be very helpful for our purposes, because we were able to overcome some limitations of relational database management systems (Demuth & Hussmann, 1999). The implementation of the UML repository, i.e., the mapping of the UML metamodel to an object-relational database schema, is described in the next section. In this section, we also outline how OCL constraints can be mapped to SQL constraints. The usage of OCL in our project is not limited to global integrity constraints. We also exploit OCL for enforcing project-related design guidelines. A short classification of guidelines with examples and their implementation as SQL constraints is given in the third section. The sample constraints illustrated in the second and third sections are manually mapped from OCL to SQL. In order to provide adequate tool support for our approach we developed an OCL-to-SQL compiler for automatic mapping of OCL constraints to SQL. This compiler is outlined in the fourth section. The final section concludes the chapter.