Chapter 3.18
UB2SQL:
A Tool for Building Database Applications Using UML and B Formal Method

Amel Mammar
University of Luxembourg, Luxembourg

Régine Laleau
University of Paris 12, France

ABSTRACT

UB2SQL is a tool for designing and developing database applications using UML and B formal method. The approach supported by UB2SQL consists of two successive phases. In the first phase, with the design of applications using class, state and collaboration diagrams, B specifications are automatically generated from UML diagrams; the diagrams are then augmented with these B specifications in place. The second phase deals with the refinement of these B specifications into a relational database implementation, for which UML representation is constructed. In both phases, proofs are achieved to ensure correctness of the obtained B specification and correctness of the refinement process. To overcome the lack of rules and tactics in the B prover, UB2SQL defines specific rules and tactics making the proof task seem like a push-button activity. To increase the usability of UB2SQL in both academic and industrial contexts, the tool has been integrated as a plug-in to the Rational Rose CASE tool. Such integration allows users to develop and be able to visualize graphical UML diagrams and formal B notation in a single environment.

INTRODUCTION

In the area of database specification and design, most research work has been dedicated to data modeling through the definition of the Entity-Relationship (ER) model and its variants. Formal rules are designed to translate ER diagrams into database schemas and included as part of the

functionality of existing industrial CASE tools. This principle has not yet been applied to database transaction design, even though some research work in that direction has been carried out (Barros, 1998; Edmond, 1995; Gunther, Schewe, & Wetzel, 1993). A couple of reasons for this should be mentioned. First, in this kind of application, data are the core component and their modeling requires much attention (since the quality of the built system mainly depends on this modeling). Second, specifying database transactions at the same abstraction level as the ER model requires the use of formal specification notations such as B (Abrial, 1996), VDM (Jones, 1990) or Z (Spivey, 1992); however, the use of formal methods raises some difficulties, as pointed out in a survey performed on experienced formal methods users (Snook & Butler, 2001). The survey observes that reading and understanding formal specifications is not a significant problem with suitable training; the main difficulty lies in creating formal specifications, and more precisely in finding useful abstractions like choosing the objects that make up the model. The authors conclude by recommending that formal methods should be more closely integrated with classical design methods, such as UML (OMG, 2003; Siau & Cao, 2001; Siau & Lee, 2004; Siau, Erickson, & Lee, 2005), which have been successfully used to design programs, and that graphical modeling tools supporting such integrations should be developed. Thus, a number of research projects (Bruel & France, 1996; Dupuy, Ledru, & Chabre-Peccoud, 2000; Ledang, Souquieres, & Charles, 2003; Marcano & Levy, 2002; Snook & Butler, 2001) have been devoted to such purpose. Formal proofs are another theme in formal methods. The objective is to increase the degree of proof-automation. A possible approach is to develop domain-dedicated provers and tools (Bernard, Legeard, Luck, & Peureux, 2004).

Our research work is at the junction of these themes and our aim is to define a method for the specification and design of database applications supported by a graphical modeling tool. In our proposal, data are first specified using UML class diagrams; transactions are then modeled with state and collaboration diagrams augmented with formal semantics. These adapted UML diagrams are converted into B specifications, which are then refined into a relational database implementation. The B method (Abrial, 1996) is a safe technique that covers the different phases of the software development life cycle: abstract specification, design by successive refinement steps and executable code generation. Using the B method, database engineers are able to check that transactions preserve integrity constraints expressed in the abstract specification. With the integration of UML+B, we can take advantage of the visual and structuring aspects of UML graphical notation, together with the rigorous reasoning possibilities of B formal notation. This paper focuses on the construction of the tool, *UB2SQL*, which supports our approach (UML+B). We highlight the main functionalities of *UB2SQL*, how it is integrated as a plug-in to the Rose CASE tool (Rational, 2003) and list the issues faced during its construction. Finally, we conclude by giving some quantitative elements and pointing out the advantages of such a tool. More details on the method itself can be found in our previous work (Laleau & Mammar, 2000a, 2000b, 2005; Mammar, 2002).

The paper is organized as follows. We start with a brief description of the B method, followed by an overview of our approach for the development of database applications and of the main functionalities of *UB2SQL*, which we describe in detail through a running example. After that, we discuss the benefits expected from such a tool, the main difficulties faced during its construction, and we compare it with other similar tools. Finally, we conclude and outline some future work.

OVERVIEW OF THE B METHOD

This section briefly presents the main concepts of the B formal method to help the reader under-