Chapter XI

Supplementing UML with Concepts from ORM

Terry Halpin
Microsoft Corporation, USA

The Unified Modeling Language (UML) is useful for designing object-oriented code, but is less suitable for conceptual data analysis. Its process-centric use-cases provide an inadequate basis for specifying data-centric class diagrams, and the UML graphical language suffers from incompleteness, inconsistency and unnecessary complexity. For example, multiplicity constraints can lead to unexpected problems when extended to n-ary associations, the constraint primitives are not optimized for orthogonality or expressibility, and the graphical language does not lend itself readily to verbalization and multiple instantiation for validating models with domain experts. This chapter examines some of these defects, and shows how to compensate for them by augmenting UML with concepts and techniques from the Object Role Modeling (ORM) approach. It highlights the potential of “data use cases” for seeding the data model, using verbalization of facts and rules with positive and negative examples to facilitate validation of business rules. The following approaches are suggested as possible ways to exploit the benefits of fact-orientation: use ORM for the conceptual analysis then map the ORM model to UML; use UML supplemented by informal population diagrams and user-defined constraints; enhance the UML metamodel to better support business rules.

INTRODUCTION

Adopted in 1997 by the Object Management Group (OMG) as a language for object-oriented (OO) analysis and design, the Unified Modeling Language (UML) has become popular for designing OO program code. It is well suited for this purpose, covering both data and behavioral aspects, and allowing OO-implementation details to be directly expressed (e.g. attribute visibility and directional navigation across associations). Although not yet widely used for designing database applications, UML can be used for this task also, since
its class diagrams effectively provide an extended Entity-Relationship (ER) notation that can be annotated with database constructs (e.g., key declarations).

Late in 1999, UML version 1.3 was approved and work began on version 1.4, with a major revision (2.0) planned for some years later. Though not yet a standard, UML has been proposed for standardization by the International Standards Organization (ISO), and approval seems likely by 2001 (Kobryn, 1999). Further background on UML may be found in its specification (OMG, 1999), a simple introduction (Fowler, 1997) or a detailed treatment (Booch et al., 1999; Rumbaugh et al., 1999). In-depth discussions of UML for database design are provided by Muller (1999) and, with a slightly different notation, by Blaha and Premerlani (1998).

The UML notation includes hundreds of symbols, from which various diagrams may be constructed to model different perspectives of an application (e.g., use case diagrams, class diagrams, object diagrams, statecharts, activity diagrams, sequence diagrams, collaboration diagrams, component diagrams and deployment diagrams). This chapter focuses on conceptual data modeling, so considers only the static structure (class and object) diagrams. Class diagrams are used for the data model, and object diagrams provide a limited means to discuss data populations.

Although useful for designing object-oriented code, UML is less suitable for developing and validating a conceptual data model with domain experts. Its use-cases are process-centric, and in practice the move from use cases to class diagrams is often little more than a black art. Moreover, incompleteness in the UML notation prevents many common business rules from being diagrammed.

It is our belief that these defects are best remedied by using fact-oriented modeling as a precursor to object-oriented modeling in UML. Object-Role Modeling (ORM) is the main exemplar of the fact-oriented approach, and though less popular than UML, is used productively in many countries and is supported by CASE tools from a number of companies, including Ascaris and Microsoft. For data modeling purposes, ORM’s graphical notation is more expressive and orthogonal than UML’s, its models and queries are more semantically stable, and its design procedures fully exploit the potential of data examples using both verbalization and multiple instantiation to help capture and validate business rules with domain experts.

This chapter identifies various flaws in the UML graphical language and discusses how fact-orientation can be used to augment the object-oriented approach of UML. It shows how verbalization of facts and rules, with positive and negative examples, facilitates validation of business rules, and compares rule visualizations in UML and ORM on the basis of specified modeling language criteria. The following three approaches are suggested as possible ways to exploit the benefits of fact-orientation: (1) use ORM for the initial conceptual information analysis and map the ORM model to a UML class diagram; (2) use UML in its current form, supplemented by informal population diagrams and user-defined constraints; (3) correct and extend the UML metamodel to better support the specification and validation of business rules.

The rest of this chapter is structured as follows. The next section provides a comparative overview of UML class diagrams and ORM, based on linguistic design criteria. The following section discusses verbalization issues related to multiplicity constraints on binary associations. The subsequent section illustrates how “data use cases” can be used to guide the data modeling process as a joint activity between modeler and domain expert. In so doing, it also exposes problems with UML multiplicity constraints on n-ary associations, and highlights the need for a richer graphical constraint notation. The conclusion summa-
Related Content

A Logic Programming Perspective on Rules
[www.irma-international.org/chapter/logic-programming-perspective-rules/35860/](www.irma-international.org/chapter/logic-programming-perspective-rules/35860/)

Native XML Programming: Make Your Tags Active
[www.irma-international.org/chapter/native-xml-programming/27781/](www.irma-international.org/chapter/native-xml-programming/27781/)

Content-Based Publish/Subscribe for XML Data
[www.irma-international.org/chapter/content-based-publish-subscribe-xml/41506/](www.irma-international.org/chapter/content-based-publish-subscribe-xml/41506/)

Version Control of Software Models
[www.irma-international.org/chapter/version-control-software-models/4930/](www.irma-international.org/chapter/version-control-software-models/4930/)

UML Modeling Support for Early Reuse Decisions in Component-Based Development
[www.irma-international.org/chapter/uml-modeling-support-early-reuse/30572/](www.irma-international.org/chapter/uml-modeling-support-early-reuse/30572/)