Augmenting UML with Fact-orientation

Terry Halpin
Microsoft Corporation, USA
TerryHa@microsoft.com

Abstract
The Unified Modeling Language (UML) is more useful for object-oriented code design than conceptual information analysis. Its process-centric use-cases provide an inadequate basis for specifying class diagrams, and its graphical language is incomplete, inconsistent and unnecessarily complex. For example, multiplicity constraints on n-ary associations are problematic, the constraint primitives are weak and unorthogonal, and the graphical language impedes verbalization and multiple instantiation for model validation. This paper shows how to compensate for these defects by augmenting UML with concepts and techniques from the Object Role Modeling (ORM) approach. It exploits "data use cases" to seed the data model, using verbalization of facts and rules with positive and negative examples to facilitate validation of business rules, and compares rule visualizations in UML and ORM. Three possible approaches are suggested: use ORM for conceptual analysis then map to UML; supplement UML with population diagrams and user-defined constraints; enhance the UML metamodel.

1. Introduction

The Unified Modeling Language (UML) was adopted in 1997 by the Object Management Group (OMG) as a language for object-oriented (OO) analysis and design. This paper is concerned with UML version 1.3, the latest approved version at the time of writing. A minor revision (1.4) should be approved around December 2000, and a major revision (2.0) should be completed a few years later. Though not yet a standard, UML has been proposed for standardization by the International Standards Organization, with approval likely around 2001 [28].

The UML notation includes the following kinds of diagram for modeling different perspectives of an application: use case diagrams, class diagrams, object diagrams, statecharts, activity diagrams, sequence diagrams, collaboration diagrams, component diagrams and deployment diagrams. This paper 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 for data populations. Although not yet widely used for designing database applications, UML class diagrams effectively provide an extended Entity-Relationship (ER) notation that can be annotated with database constructs (e.g. key declarations). Background on UML may be found in its specification [31], a simple introduction [13] or a detailed treatment [6, 32]. In-depth discussions of UML for database design may be found in [30] and (with a slightly different notation) [3].

UML has become popular for designing OO program code. It is well suited for this purpose, covering data, behavior, and OO-implementation details (e.g. attribute visibility and directional navigation across associations). However, 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, the UML notation prevents many common business rules from being diagrammed.

We believe these defects are best avoided 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 is supported by CASE tools such as Microsoft Visio Enterprise [34]. For data modeling, ORM’s graphical notation is more expressive and orthogonal than UML’s, its models and queries are semantically stabler, and its design procedures fully exploit data examples using both verbalization and multiple instantiation to help capture and validate business rules with domain experts.

This paper identifies several weaknesses in the UML graphical language and discusses how fact-orientation can 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 conceptual information analysis and map the ORM model to UML; (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 business rules.