IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITB7418

Chapter XVI

Information Analysis in UML and ORM: A Comparison

Terry Halpin
Microsoft Corporation, USA

Since its adoption by the Object Management Group as a language for object-oriented analysis and design, the Unified Modeling Language (UML) has become widely used for designing object-oriented code. However, UML has had only minimal adoption among practitioners for the purposes of information analysis and database design. One main reason for this is that the class diagrams used in UML for data modeling provide only weak, and awkward, support for the kinds of business rules found in data-intensive applications. Moreover, UML's graphical language does not lend itself readily to verbalization and multiple instantiation for validating data models with domain experts. These defects can be remedied by using a fact-oriented approach for information analysis, from which UML class diagrams may be derived. Object-Role Modeling (ORM) is currently the most popular fact-oriented modeling approach. This chapter examines the relative strengths and weaknesses of UML and ORM for conceptual data modeling, and indicates how models in one notation can be translated into the other.

INTRODUCTION

The *Unified Modeling Language* (UML) was adopted by the Object Management Group (OMG) in 1997 as a language for object-oriented (OO) analysis and design. At the time of writing, the latest approved specification for UML is version 1.4 (OMG, 2001), with various proposals for version 2.0 under consideration. Our discussion of UML is confined to version 1.4. For a simple introduction to UML, see Fowler (1997). Detailed treatments of UML are provided by Booch, Rumbaugh, and Jacobson (1999), and Rumbaugh, Jacobson, and Booch (1999). In-depth discussions of UML for database design are given by Muller (1999) and, with a slightly different notation, by Blaha and Premerlani (1998).

UML can be used for this task also, since a class diagram can be viewed as an extended Entity-Relationship (ER) notation, assuming it is annotated with user-defined constraints for database constructs (e.g., key declarations).

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, state-charts, 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.

UML is suitable for OO code design, covering both data and behavioral aspects, and allowing OO-implementation details to be directly expressed (e.g., attribute visibility and directional navigation across associations). UML may be also used for analysis by ignoring its implementation features. Although not widely used for modeling database applications, analysis versions of UML class diagrams effectively provide an Entity-Relationship (ER) notation, assuming they are annotated with additional user-defined constraints (e.g., uniqueness declarations for attributes).

In practice, however, UML static structure diagrams are often unsuitable for information analysis, since they can make it awkward to capture and validate data concepts and business rules with domain experts, and to cater for structural changes in the application domain. These problems can be remedied by using a fact-oriented approach for information analysis, where communication takes place in simple sentences; each sentence type can easily be populated with multiple instances, and attributes are eschewed in the base model. At design time, a fact-oriented model can be used to derive a UML class model or a logical database model.

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 companies such as Ascaris and Microsoft. ORM harmonizes well with UML, since both approaches provide direct support for association roles, n-ary associations and objectified associations. ORM pictures the world simply in terms of objects (entities or values) that play roles (parts in relationships). For example, you are now playing the role of reading, and this chapter is playing the role of being read. Overviews of ORM may be found in Halpin (1998a, 1998b) and a detailed treatment in Halpin (2001a).

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. These design principles are then used to examine the relative strengths and weaknesses of UML and ORM for information analysis, focusing first on data structures, and then moving on to constraints, outlining how models in one notation can be translated into the other. An example of schema transformation and optimization is then used to underscore the importance of constraint expressibility. We then briefly compare textual language support for constraints, derivation rules and queries in the two approaches. The conclusion summarizes the main points and identifies topics for future research.

CONCEPTUAL MODELING LANGUAGE CRITERIA

A modeling method comprises both a language and a procedure to guide modelers in using the language to construct models. A language has associated syntax (marks), semantics