Chapter XIII

Use of UML Stereotypes in Business Models

Daniel Brandon, Jr.
Christian Brothers University, USA

ABSTRACT

This chapter presents some particularly useful UML stereotypes for use in business systems. Stereotypes are the core extension mechanism of UML. If you find that you need a modeling construct that isn’t in the UML but it is similar to something that is, you treat your construct as a stereotype. These new stereotypes are defined and the need for them is discussed. The stereotypes aid in both the design/drawing phase and the implementation (coding) phase of the overall system construction. An example case study illustrates their usage with both design drawings and the implementation code (C++).

INTRODUCTION

The UML (Unified Modeling Language) has become a standard in software design of object-oriented computer systems. UML provides for the use of stereotypes to extend the utility of its base capabilities. In the design and construction of business systems, we have found some particularly useful stereotypes, which are defined and illustrated in this chapter.

BACKGROUND

“Stereotypes are the core extension mechanism of UML. If you find that you need a modeling construct that isn’t in the UML but it is similar to something that is, you treat your construct as a stereotype” (Fowler, 2000). The stereotype is a semantic added to an existing model element and when diagrammed, it consists of the stereotype name inside of guillemots (a.k.a. chevrons) within the selected model’s element. The guillemot
looks like a double angle bracket (<< ... >>), but it is a single character in extended font libraries (Brown, 2002). The UML defines about 40 of these stereotypes such as “<<becomes>>,” “<<include>>,” and “<<signal>>” (Scott, 2001). However, these 40 standard stereotypes do not add the necessary meaning for automatic code generation in a UML case tool.

One common, general use of the stereotype is for a meta-class. A meta-class is a class whose instances are classes; these are typically used in systems in which the developer needs to declare classes at run time (Eriksson, 1998). A similar general use is for powertypes. A powertype is an object type (class) whose instances are sub-types of another object type. Figure 1 shows an example of the use of stereotypes for powertypes (Martin, 1998).

**USER-DEFINED STEREOTYPES FOR BUSINESS SYSTEMS**

In the design of business systems, we have found some stereotypes that were useful, and two stereotypes that were extremely useful. When defining stereotypes, it is necessary to describe (Eriksson, 1998):

1. On which [UML] element the user-defined stereotype should be based.
2. The new semantics the stereotype adds or refines.
3. One or more examples of how to implement the user-defined stereotype.

A common use of stereotypes in business systems is for interfaces as found in Java or CORBA; this is shown in Figure 2. An interface typically has public functionality but not data (unless holding data for global constants). The class model element has been modified with the “<<interface>>” notation. This is commonly used for UML case products that do not have separate interface symbols or where these symbols do not allow data (i.e., global constants).

Still another common use of a stereotype in business systems is to clarify or extend a relationship. Figure 3 shows a stereotype called “history” which implies a “many” cardinality for history purposes, i.e., each person has zero or one current employer(s), but may have many employers in terms of the employee’s history. It may imply some common functionality upon code generation such as (Fowler, 2000):

Company Employee::getCompany(Date);