Chapter I
Enriched Conceptualization of Subtyping

Terry Halpin
Neumont University, USA

ABSTRACT

When modeling information systems, one often encounters subtyping aspects of the business domain that can prove challenging to implement in either relational databases or object-oriented code. In practice, some of these aspects are often handled incorrectly. This chapter examines a number of subtyping issues that require special attention (e.g. derivation options, subtype rigidity, subtype migration), and discusses how to model them conceptually. Because of its richer semantics, the main graphic notation used is that of second generation Object-Role Modeling (ORM 2). However, the main ideas could be adapted for UML and ER, so these are also included in the discussion. A basic implementation of the proposed approach has been prototyped in Neumont ORM Architect (NORMA), an open-source tool supporting ORM 2.

INTRODUCTION

In the wider sense, an information system corresponds to a business domain or universe of discourse rather than an automated system. As the name suggests, the universe of discourse is the world, or context of interest, about which we wish to discourse or talk. Most business domains involve some subtyping, where all instances of one type (e.g. Manager) are also instances of a more encompassing type (e.g. Employee). In this example, Manager is said to be a subtype of Employee (a supertype).

Various information modeling approaches exist for modeling business domains at a high level, for example Entity-Relationship Model-

ing (ER) (Chen, 1976), the Unified Modeling Language (UML) (Object Management Group, 2003a, 2003b; Rumbaugh, Jacobson & Booch, 1999), and Object-Role Modeling (ORM) (Halpin, 2006, 2007; Halpin & Morgan, 2008). These modeling approaches provide at least basic subtyping support. In industrial practice however, certain aspects of subtyping are often modeled or implemented incorrectly. This is sometimes due to a lack of appropriate modeling constructs (e.g. derivations to/from subtypes, subtype rigidity declarations), or to a lack of an obvious way to implement a subtyping pattern (e.g. historical subtype migration). This paper proposes solutions to some of these issues. Because of its richer semantics, the main graphic notation used is that of ORM 2 (second generation ORM), as implemented in NORMA, an open source ORM 2 tool. However, the main ideas could be adapted for UML and ER.

The next section overviews basic subtyping and its graphical depiction in ORM, UML, and ER, and identifies the condition under which formal derivation rules are required. The section after that proposes three varieties of subtyping (asserted, derived, and semiderived). The subsequent section distinguishes rigid and role subtypes, relates them to changeability settings on fact type roles, and discusses a popular party pattern. The next section discusses various patterns for modeling history of subtype or role migration. The final section notes implementation issues, summarizes the main results, suggests future research topics, and lists references.

BASIC SUBTYPING AND THE NEED FOR DERIVATION RULES

Figure 1(a) shows a simple case of subtyping in ORM 2 notation. Patients are identified by their patient numbers and have their gender recorded. Patient is specialized into MalePatient and FemalePatient. Pregnancy counts are recorded for, and only for, female patients. Prostate status is recorded only for male patients. In ORM 2, object types (e.g. Patient) are depicted as named, soft rectangles. A logical predicate is depicted as a named sequence of role boxes, each connected by a line to the object type whose instances may play that role. The combination of a predicate and its object types is a fact type—the only data structure in ORM (relationships are used instead of attributes). If an object type is identified by a simple fact type (e.g. Gender has GenderCode) this may be abbreviated by placing the reference mode in parentheses.

A bar spanning one or more roles depicts a uniqueness constraint over those roles (e.g. **Each** Patient has **at most one** Gender). A large dot depicts a mandatory constraint (e.g. **Each** Patient has **some** Gender). The circled dot with a cross through it depicts an exclusive-or constraint (**Each** Patient **is a** MalePatient **or is a** FemalePatient **but not both**). Overviews of ORM may be found in Halpin (2005b, 2006, 2007), a detailed treatment in Halpin & Morgan (2008), and a metamodel comparison between ORM, ER, and UML in Halpin (2004). Various dialects of ORM exist, for example Natural language Information Analysis Method (NIAM) (Wintraecken, 1990) and the Predicator Set Model (PSM) (ter Hofstede et al., 1993).

Figure 1(b) shows the same subtyping arrangement in UML. In UML, the terms "class" and "subclass" are used instead of "object type" and "subtype"". The "{P}" notation is the author's nonstandard addition to UML to indicate that an attribute is (at least part of) the preferred identifier for instances of the class. ORM and UML show subtypes outside their supertype(s), and depict the "is-a" relationship from subtype to supertype by an arrow.

The Barker ER notation (Barker, 1990), arguably the best industrial ER notation, instead uses an Euler diagram, placing the subtype shapes within the supertype shape, as shown in Figure 1(c). In spite of its intuitive appeal, the Barker ER subtyping notation is less expressive than that