Temporal Modeling: Part 5

Terry Halpin
Neumont University

This is the fifth in a series of articles on the impact of *time* on the conceptual modeling of business domains. The first article [8] discussed the temporal data types *instant* (point in time), *interval* (duration of time), and *period* (anchored duration of time), classified *temporal object types* into *once-only* (e.g., Date) and *repeatable* (e.g., WeekDay) object types, and discussed *four kinds of fact type*: *definitional* (truth of instances is a matter of definition), *once-only* (instances correspond to a single event), *repeatable* (instances may correspond to multiple events) and *time-deictic* (the meaning of instances depends on the time of utterance/inscription). It then showed how to model temporal details about point events or period events underlying instances of once-only fact types that are unchangeable. The second article [9] examined the modeling of temporal information about events underlying *changeable fact types* (their fact populations may change over time, by replacing, adding, or deleting facts) that are *functional* (*n*:1 or 1:1 associations). The third article [10] discussed how to maintain history of *changeable fact types* that are *nonfunctional* (e.g. *m:n* binaries, or higher arity fact types). The fourth article [11] provided another way in UML 2 to maintain history of nonfunctional, changeable fact types, and then discussed *rigid subtypes and role subtypes,* and related dynamic constraints.

This fifth article discusses one way to maintain history of objects as they migrate from one role subtype to another. Three graphical notations are used for examples: second generation Object-Role Modeling (ORM 2) [5, 6] as supported by the open source (Neumont ORM Architect) NORMA tool [3, 13]; the Unified Modeling Language (UML) [14]; and the Barker notation [1] for Entity-Relationship Modeling (ER) [2].

Subtype Migration

In the previous article [11], I distinguished between rigid types and role types. Instances of a *rigid type* (e.g. Person, Book) necessarily remain in that type for their whole lifetime. In contrast, instances of *role types* (e.g. Employee, Judge) may move into or out of those types at different times during their lifetime. If an object type has multiple subtypes that are role types, *migration* between these role subtypes is often allowed. For example, Figure 1 depicts three role subtypes of Person in (a) ORM, (b) UML, and (c) Barker ER notation. Although some researchers classify these subtypes as phases rather than roles (e.g. [4]), I use the term "role" more liberally to include phases. Hence Child, Teenager and Adult are treated as different roles that a person may take on. In each of these schemas, the role subtypes are declared to be mutually exclusive. ORM depicts this constraint with a circled "X" (for "eXclusion"), UML uses "{disjoint}", and Barker ER always assumes that an entity type's subtypes are exclusive.

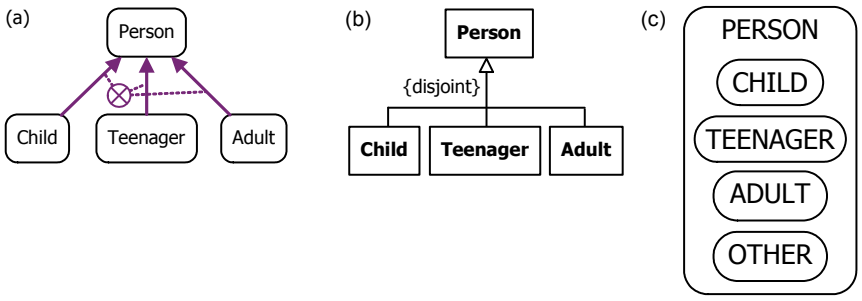

Figure 1 Example of mutually exclusive role subtypes in (a) ORM, (b) UML, and (c) Barker ER.

The role subtypes in Figure 1 are not collectively exhaustive, because we allow that some people (e.g. babies) might not belong to any of these roles at some time in their lifetime. This is shown by the absence of an exhaustion constraint (circled dot in ORM, {complete} in UML, and the inclusion of Other in Barker ER). If we were instead to use the term "Child" more liberally to include all phases prior to teenagerhood, then we would have a partition, and an exhaustion constraint would need to be added.

The exclusion constraint in Figure 1 is a *static* constraint, and hence applies to each state of the model, taken individually. In other words, at no time can a person simultaneously play more than one of the roles shown. Over time however, a person may move from one of these roles to another. With this example, there is a strict *linear ordering* to this *subtype migration*, as shown by the state chart in Figure 2.

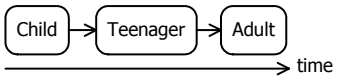

Figure 2 A state chart indicating the possible transitions between the roles in Figure 1.

Now suppose that we are interested in recording facts about people at different phases of their life. For example, suppose we wish to record a person's favorite toy (if any) when he/she was a child, a person's favorite pop group (if any) when he/she was a teenager, and a person's favorite book (if any) when he/she was an adult. Is it OK to simply add these fact types to our previous schema, as shown in Figure 3?

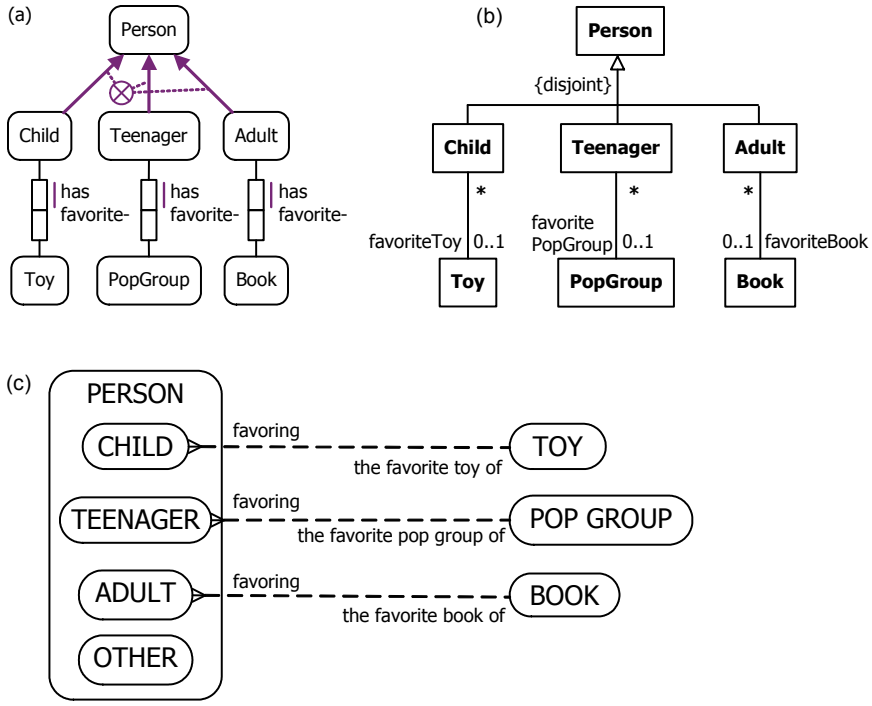

Figure 3. Adding subtype specific details in (a) ORM, (b) UML and (c) Barker ER.

This is acceptable if we agree to record personal details only for the most recent role played, but it is not acceptable if we want to remember details about a person for more than one role. Since a person can be playing only one of these roles at a time, if we use this schema to record an adult person's favorite book, we cannot also record his/her favorite popgroup as a teenager or his/her favorite toy as a child. But suppose we *do* want to retain a history of such details over a person's lifetime, as he/she moves from one role to another. The next section discusses one way to address this need to model *history of subtype migration*. The next article will discuss another way to model this requirement.

Maintaining History of Subtype Migration with the Decreasing Disjunctions Pattern

I first ran into this kind of problem over twenty years ago, when I had to model history of people who applied for entry into a boy's school, possibly later became students at the school, and possibly later left the school and joined their old boys association. Details were to be maintained for each stage. For example, when asking old boys to donate to the school, it was helpful to be able to recount to them some of their past achievements while they were students at the school. I call the solution I came up with then the *decreasing disjunctions pattern*, since it has a top supertype that disjoins all the roles, then successively subtypes to smaller disjunctions. I'm using "disjunction" here to mean exclusive, logical disjunction (exclusive-or).

With this example, we could have called the top supertype "Child or Teenager or Adult", but if we are not interested in recording specific details for other roles (e.g. Baby) then it's simpler to use the original supertype (i.e. Person). Figure 4 illustrates this approach in (a) ORM, (b) UML, and (c) Barker ER notations. Clearly, this enables us to record and remember details of a person for all three phases (child, teenager, and adult).

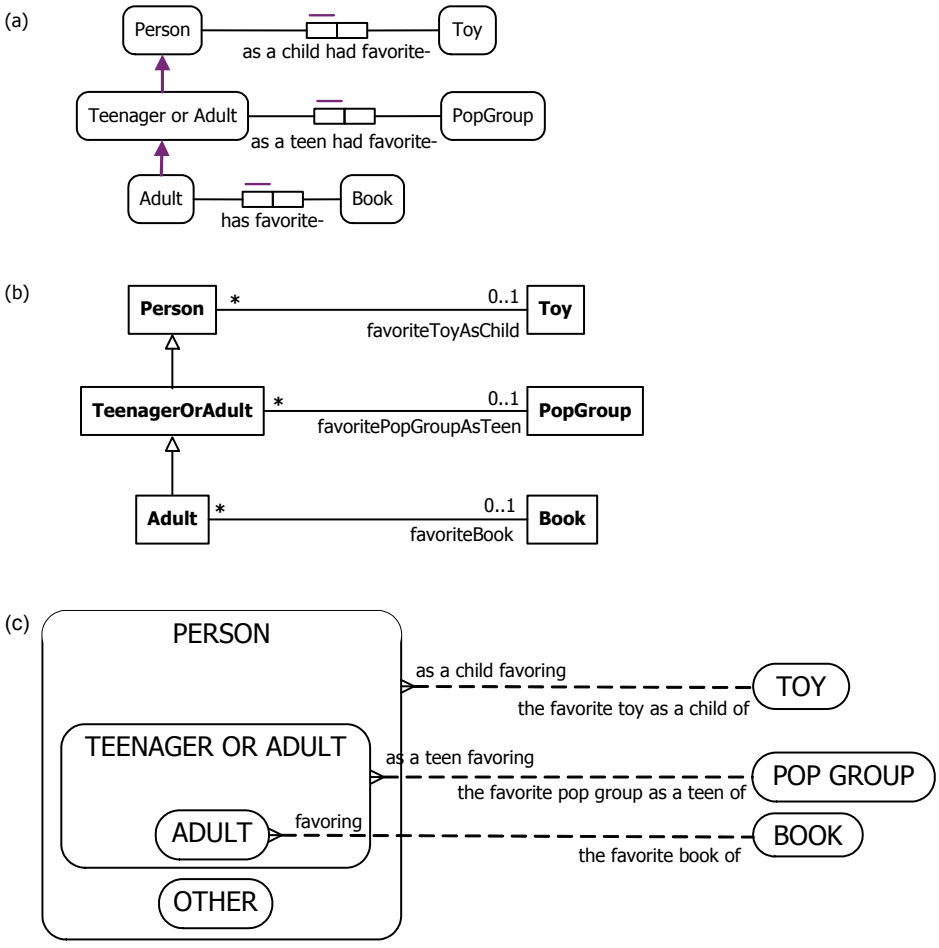

Figure 4. Applying the decreasing disjunctions patterns to retain history of subtype migration.

Conclusion

One advantage of the decreasing disjunctions pattern is that the linear order of the role transitions is effectively enforced by the subtyping order. However this becomes a disadvantage if the role transition is not linear (e.g. if a role may be repeatedly played more than once during an object's lifetime). For example, a person may play the role of being married, then the role of being divorced or widowed, and then play the role of being married again. If we want to retain a history of specific details for nonlinear cases like this, another data model pattern is needed. The next article in this series discussed different patterns for handling history of subtype migration for both linear and nonlinear state transitions.

References

1. Barker, R. 1990, *CASE*Method: Tasks and Deliverables*, Addison-Wesley, Wokingham, England.
2. Chen, P. P. 1976, 'The entity-relationship model—towards a unified view of data'. *ACM Transactions on Database Systems*, 1(1), pp. 9–36.
3. Curland, M. & Halpin, T. 2007, 'Model Driven Development with NORMA', *Proc. 40th Int. Conf. on System Sciences (HICSS-40)*, 10 pages, CD-ROM, IEEE Computer Society, January 2007.
4. Guizzardi, G. 2005, Ontological Foundations for Structural Conceptual Models, CTIT PhD Thesis Series, No. 05-74, Enschede, The Netherlands.
5. Halpin, T. & Morgan, T. 2008, *Information Modeling and Relational Databases*, *2nd edition*, Morgan Kaufmann, San Francisco.
6. Halpin, T. 2005, 'ORM 2', *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, eds. R. Meersman, Z. Tari, P. Herrero et al., Cyprus. Springer LNCS 3762, pp 676-87.
7. Halpin, T. 2006, 'Verbalizing Business Rules: Part 14', *Business Rules Journal*, Vol. 7, No. 4 (April 2006), URL: http://www.BRCommunity.com/a2006/b283.html.
8. Halpin T. 2007, 'Temporal Modeling (Part 1)', *Business Rules Journal*, Vol. 8, No. 2 (Feb. 2007), URL: http://www.BRCommunity.com/a2007/b332.html.
9. Halpin, T. 2007, 'Temporal Modeling (Part 2)', *Business Rules Journal*, Vol. 8, No. 6 (June 2007), URL: http://www.BRCommunity.com/a2007/b351.html.
10. Halpin, T. 2007, 'Temporal Modeling (Part 3)', *Business Rules Journal*, Vol. 8, No. 11 (Nov. 2007), URL: http://www.BRCommunity.com/a2007/b374.html.
11. Halpin, T. 2008, 'Temporal Modeling (Part 4)', *Business Rules Journal*, Vol. 9, No. 4 (Apr. 2008), URL: http://www.BRCommunity.com/a2008/b411.html.
12. Halpin, T. & Proper, H. 1995, 'Subtyping and polymorphism in object-role modelling', *Data & Knowledge Engineering*, vol. 15, no. 3, pp. 251–281.
13. NORMA website: http://www.ormfoundation.org and http://sourceforge.net/projects/orm.
14. Object Management Group 2003, *UML 2.0 Infrastructure*, URL: http://www.omg.org/uml.