Temporal Modeling: Part 6

Terry Halpin
Neumont University

This is the sixth 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. The fifth article [12] introduced the *decreasing disjunctions pattern* to maintain history of *migration between subtypes* when the state transition graph is linear.

This sixth article discusses an alternative way to maintain history of objects as they migrate from one role subtype to another, for linear state transition cases: the *once-only role-playing pattern*. Three graphical notations are used for examples: second generation Object-Role Modeling (ORM 2) [5, 6] as supported by the open source NORMA tool [3, 14]; the Unified Modeling Language (UML) [15]; and the Barker notation [1] for Entity-Relationship Modeling (ER) [2].

Modeling Linear Subtype Migration with the Once-only Role-playing Pattern

The previous article [12] considered the need to retain subtype-specific details of objects as they move from one role subtype to another, for linear transition cases where objects never return to play a role once they leave it. As an example, we required recording of details about one's favorite toy as a child, one's favorite pop group as a teenager, and one's favorite book as an adult. The solution provided used the decreasing disjunctions pattern, where successive subtypes remove an alternative (disjunct), as shown in Figure 1 for (a) ORM and (b) UML.

Here Person is equivalent to the disjunction Child-or-Teenager-or-Adult. Subtyping to Teenager-or-Adult removes the Child role, and subtyping further to Adult removes the Teenager role. With this model, we may record details about a person for all three phases/roles (Child, Teenager, Adult) of their life, even though at any point in time they belong to only one of these phases/roles. Recall that, unlike some other approaches [4], we use "role type" liberally to include phase types. For the same model in Barker ER notation, see [12].

As the state chart in Figure 1(c) indicates, this is a case of linear state transitions, where a person can never return to one of these phases/roles once they leave it. In this sense, we regard Child, Teenager, and Adult as *once-only roles* (roles that are never repeated, i.e. once you've played and left that role, you never play it again).

In cases like this, as an alternative to the decreasing disjunctions pattern, we may model the situation using what I call the *once-only role playing pattern*, where the playing of a once-only role is treated as an object in its own right. The life role playing type may then be subtyped into the three phases, and the relevant details for each phase attached to each.

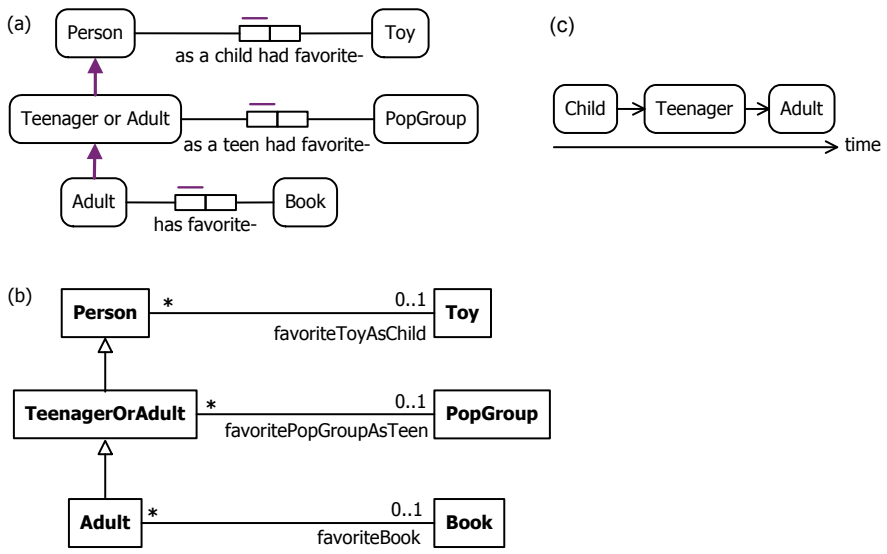

Figure 1 Applying the decreasing disjunctions pattern in (a) ORM and (b) UML for linear state transitions (c).

An ORM solution using this approach is shown in Figure 2. For completeness, reference schemes have been added as well as subtype definitions. Here the subtypes (PersonAsChild etc.) are subtypes of LifeRolePlaying (the *objectification* of Person plays LifeRole), not subtypes of Person. Like the decreasing disjunctions pattern, the once-only role playing pattern assumes that each subtype is a once-only role, but unlike the previous pattern it does not specify the linear order of the roles. In ORM 2's Fact Oriented Modeling Language (FORML), this transition order may be specified as a dynamic textual constraint on the fact type Person plays LifeRole as follows, using "**previous**" in the derivable sense of "most recent". As this is a case of adding rather than replacing facts, "**added**" is used instead of "**new**". On the ORM diagram, this dynamic constraint appears as a footnote on the constraint context (Person).

For each Person,
in case previous lifeRole =
'Child': **added** lifeRole = 'Teenager'
'Teenager': **added** lifeRole = 'Adult'
end cases.

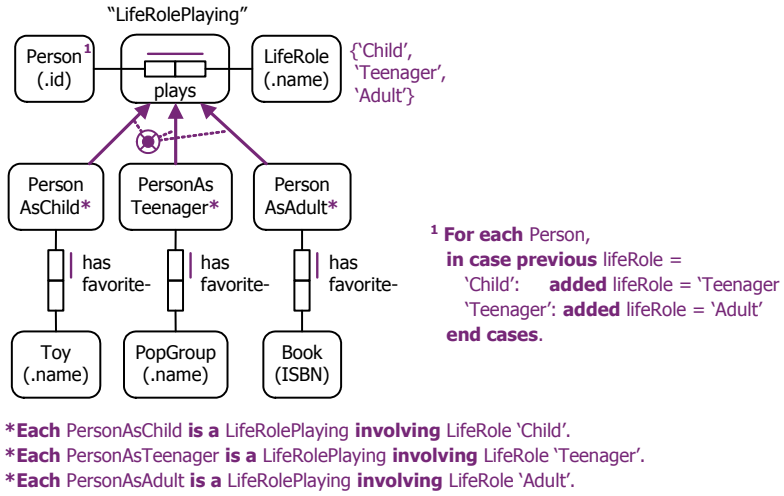

Figure 2 Applying the once-only role playing pattern in ORM.

Figure 3 shows one way of modeling the same approach in UML. Here an *association class* is used to capture the objectification LifeRolePlaying. The specific details recorded for the subclasses are modeled here as attributes. Recall that {P} is our nonstandard notation for preferred identifier. The subclass definitions are specified informally in notes. If we also wished to record something about favorite toys, pop groups or books, then these details would instead be modeled as associations to Toy, PopGroup and Book classes. The dynamic constraint on state transitions may be shown on a separate diagram, such as a state machine diagram which looks similar to Figure 1(c), without the time axis.

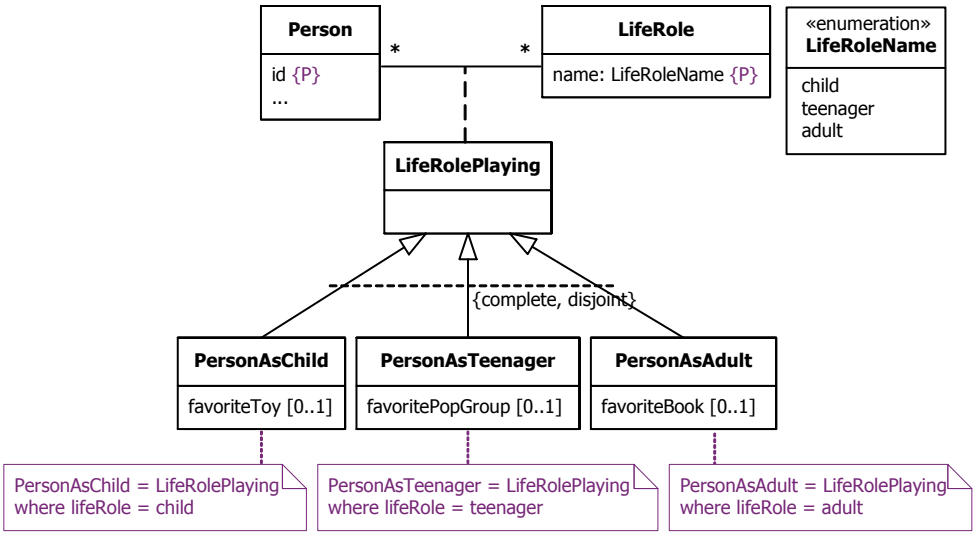

Figure 3. Applying the once-only role-playing pattern in UML.

In ORM, any objectified association may be remodeled as a *coreferenced type*, allowing a variation of the once-only role playing pattern. So the objectified association LifeRolePlaying may instead be modeled as a coreferenced type, participating in the fact types LifeRolePlaying is by Person and LifeRolePlaying is of LifeRole, as shown in Figure 4. Here the circled double-bar indicates an external uniqueness constraint (each combination of Person and LifeRole instances refers to at most one LifeRole Playing) that is used as the preferred identifier for LifeRolePlaying. For simplicity, the subtype definitions and dynamic rule are omitted (they are the same as given previously).

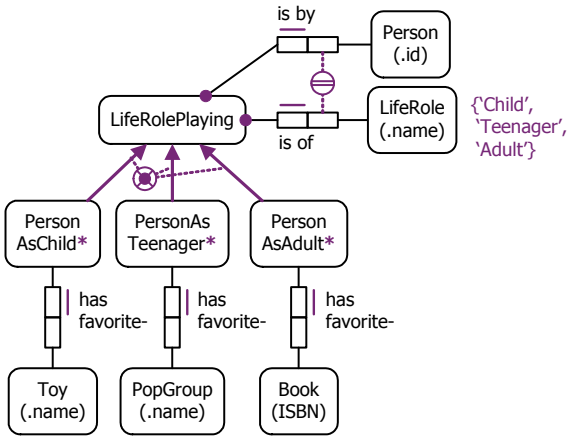

Figure 4. Applying the coreferenced version of the once-only role-playing pattern in ORM.

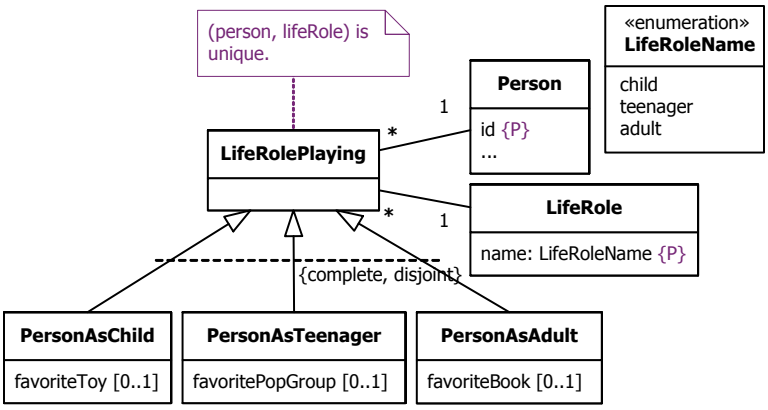

Figure 5. Applying the coreferenced version of the once-only role-playing pattern in UML.

Figure 5 shows an equivalent model in UML. Since UML lacks any graphic notation for external uniqueness, this constraint is captured informally in a note.

One advantage of the coreferenced version of the pattern is that it enables the pattern to be used in modeling approaches such as industrial ER that do not support objectified associations. For example, Figure 6 shows the basic solution in the Barker ER notation. The strokes through the association lines indicate that instances of the entity type LifeRolePlaying are identified by (person, lifeRole) pairs instantiating the far roles of these associations. As the Barker ER notation does not support subtype definitions or enumerations, much less dynamic constraints, these aspects need to be noted separately.

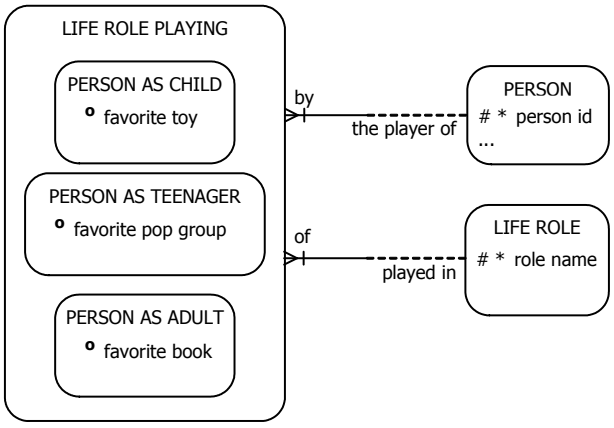

Figure 6. Applying the coreferenced version of the once-only role-playing pattern in Barker ER.

Conclusion

This article illustrated the use of the once-only role-playing pattern to model history of objects as they move from one role type to another. Both this pattern, and the decreasing disjunctions pattern considered in the previous article, are restricted to cases where the roles are once-only (once you've played and left that role, you never play it again). These two patterns cannot be used to maintain history in cases where 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. The next article discusses a third pattern, the repeatable role playing pattern, to handle such cases.

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. 2008, 'Temporal Modeling (Part 5)', *Business Rules Journal*, Vol. 9, No. 10 (Oct. 2008), URL: http://www.BRCommunity.com/a2008/b444.html.
13. Halpin, T. & Proper, H. 1995, 'Subtyping and polymorphism in object-role modelling', *Data & Knowledge Engineering*, vol. 15, no. 3, pp. 251–281.
14. NORMA website: http://www.ormfoundation.org and http://sourceforge.net/projects/orm.
15. Object Management Group 2003, *UML 2.0 Infrastructure*, URL: http://www.omg.org/uml.