Temporal Modeling: Part 7

Terry Halpin
LogicBlox

This is the seventh in a series of articles on the impact of *time* on the conceptual modeling of business domains. The first article [7] 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, 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), and then showed how to model temporal details about point events or period events underlying unchangeable instances of once-only fact types. The second article [8] 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 [9] 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 [10] 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 [11] introduced the *decreasing disjunctions pattern* to maintain history of *migration between subtypes* when the state transition graph is linear. The sixth article [12] discussed the *once-only role-playing pattern* as an alternative way to maintain history of objects as they migrate from one role subtype to another, for linear state transition cases.

The decreasing disjunctions pattern and the once-only role-playing pattern 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. This seventh article discusses a third pattern, the *repeatable role-playing pattern*, to handle such cases. Three graphical notations are used for examples: second generation Object-Role Modeling (ORM 2) [4, 5] as supported by the NORMA tool [3, 14]; the Unified Modeling Language (UML) [15]; and the Barker notation [1] for Entity-Relationship Modeling (ER) [2].

Modeling Subtype Migration with the Repeatable Role-playing Pattern

Recall that a type is a *rigid type* if each instance of it must remain in that type for the duration of that instances's lifetime (e.g. Person, Tree), otherwise the type is a *role type* (e.g. Employee, Cricketer). Over time, an entity may move from one role type to another. Suppose each role has specific details of interest and we want to maintain this history of an entity as it changes roles. We classify role subtypes as once-only or repeatable. With a once-only role subtype, objects can never return to play that role again once they have left the subtype (e.g. Child, SinglePerson). With a repeatable role subtype, objects can return to play that role again (e.g. Employee, MarriedPerson).

The two previous articles [11, 12] considered two data model patterns for retaining 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 first solution provided used the decreasing disjunctions pattern, where successive subtypes remove an alternative (disjunct). The second solution used 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.

Sometimes role playing may be *repeated*, allowing loops in the role state graph. For example, the state chart in Figure 1 specifies the allowed marital state transitions. A person may play the roles of Married, Widowed, and Divorced more than once. Many other examples of this kind occur in business domains.

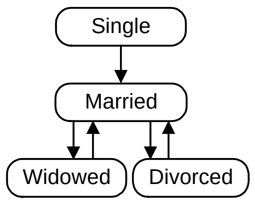
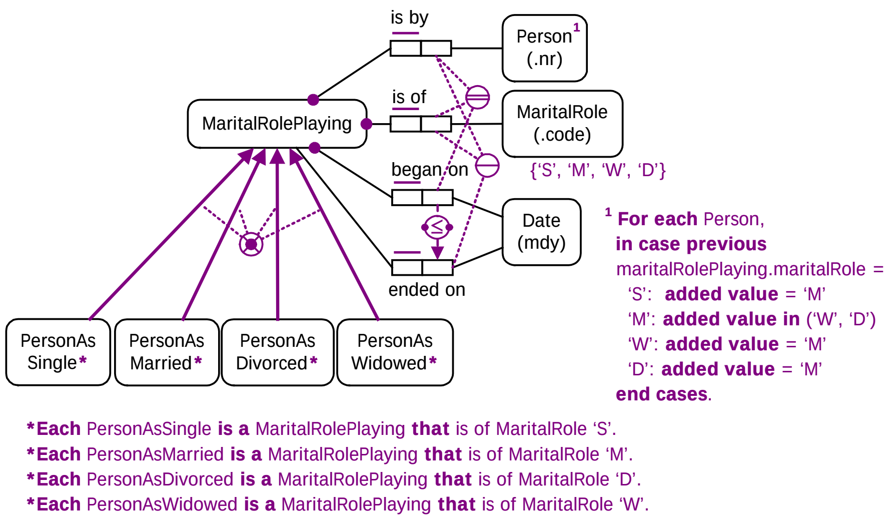


Figure 1 A state chart providing an example of repeatable roles.

If a role subtype is repeatable, the decreasing disjunctions pattern and the once-only role playing pattern cannot record history of multiple playings of the same role by the same object. To address this problem, we provide what we call the *repeatable role-playing pattern*, which includes the start-time of a role playing as part of its natural identifier. For example, the marital role playing example may be modeled in ORM as shown in Figure 2. Here the circled double-bar depicts an external uniqueness constraint enabling the preferred identification scheme for MaritalRolePlaying (the combination of person, marital role and startdate identifies the role playing). The circled single-bar is an external uniqueness constraint indicating that the combination of person, marital role and enddate (if it exists) also applies to only one marital role playing. These external uniqueness constraints assume that a person may begin or end a given marital role at most once on the same date (if this is not true, replace Date by a temporal type of finer temporal granularity to ensure the uniqueness, e.g. Instant).



[1] **For each** Person,
in case previous
maritalRolePlaying.maritalRole =
'S': **added value** = 'M'
'M': **added value in** ('W', 'D')
'W': **added value** = 'M'
'D': **added value** = 'M'
end cases.

***Each** PersonAsSingle **is a** MaritalRolePlaying **that** is of MaritalRole 'S'.
***Each** PersonAsMarried **is a** MaritalRolePlaying **that** is of MaritalRole 'M'.
***Each** PersonAsDivorced **is a** MaritalRolePlaying **that** is of MaritalRole 'D'.
***Each** PersonAsWidowed **is a** MaritalRolePlaying **that** is of MaritalRole 'W'.

Figure 2 The coreferenced version of the repeatable role-playing pattern in ORM.

The subtypes are derived, as indicated by the asterisks and subtype definitions. For example, an instance of the PersonAsMarried subtype corresponds to a given person playing an occurrence of the married role. If a person marries twice, this produces two instances of this subtype. Fact types may be attached to each of the subtypes to record details of interest, so relevant history may be maintained of a person as he/she moves from role to role (perhaps repeatedly). The dotted and circled "≤" symbol depicts the value-comparison constraint that for each marital role playing, its start date must be on or before its end date (if it exists). The textual constraint shown as footnote 1 on Person is a dynamic constraint declaring the allowed transitions between marital states.

The basic, repeatable role-playing pattern underlying concrete examples like this marriage case is set out in Figure 3, with subtyping and the transition rule omitted. The ORM model in Figure 3(a) assumes that an actor (e.g. a person, organization, or robot) may begin or end a given role at most once on the same date (if this is not true, replace Date by a finer temporal type to ensure the uniqueness, e.g. Instant). This pattern allows that an actor may begin or end multiple roles on the same date.

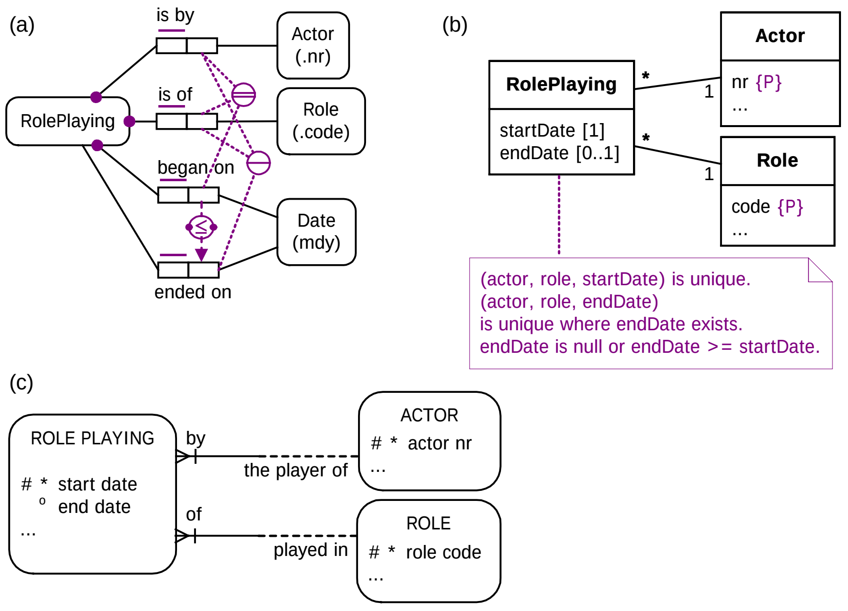


Figure 3. A basic, repeatable role-playing pattern in (a) ORM, (b) UML, and (c) Barker ER.

Figure 3(b) shows one way to model the pattern in UML. Recall that {P} is our nonstandard notation for preferred identifier. The two external uniqueness constraints and the value-comparison constraint in the ORM model are captured informally in a note—they could be formally specified in OCL, at the risk of being unintelligible for most domain experts. In UML, the dynamic constraint on state transitions may be shown in a state machine diagram that looks similar to Figure 1.

Figure 3(c) shows the basic pattern in Barker ER. The preferred identifier constraint is captured graphically by the combination of the # on start date and the stroke through the two relationships. The other external uniqueness constraint and the value-comparison constraint cannot be captured graphically in Barker ER, so should be written down somewhere separately.

Basic data model patterns for role playing using Barker ER notation have also been described by Dave Hay [13] and Len Silverston [16], but these ignore the fundamental impact of the state graph topology (e.g., whether looping is allowed) on the model requirements.

Instead of using the start date as part of the preferred identifier for the role playing type, a simple numeric identifier (RolePlayingId) may be used, as shown in Figure 4. So long as this identifier remains visibly part of human communication, it may be used even if the external uniqueness constraints do not apply. The Barker ER model loses both external uniqueness constraints, since it allows only one identifier to be declared for any entity type, and this is now taken by role playing id.

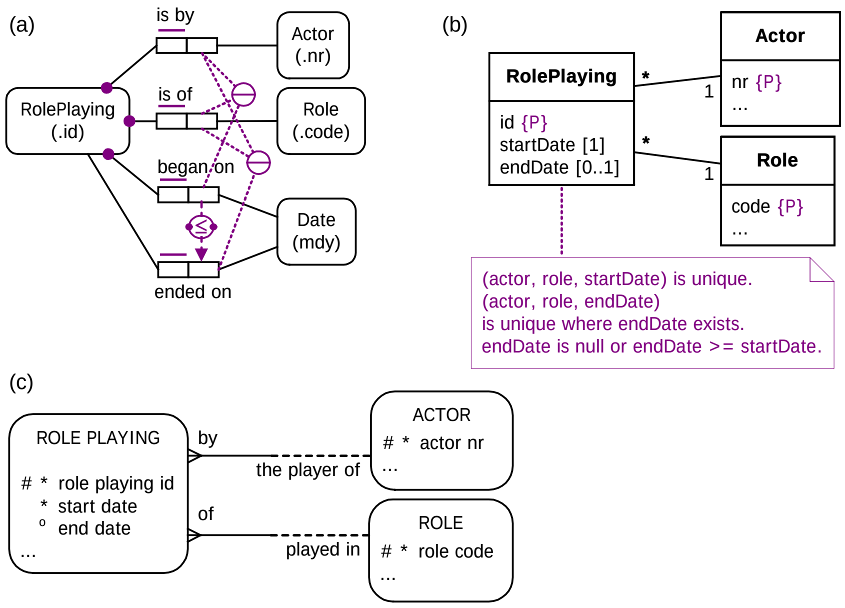


Figure 4. An alternative repeatable role-playing pattern using a simple identifier for role playing.

Yet another variation is to include an ordinal number instead of the start date as part of the identifier, as shown in Figure 5. For example, with our marriage case we might identify one particular role playing as the third (ordinal number = 3) marriage role of a given person (e.g. Elizabeth Taylor). Again, this identification scheme works even if the secondary external uniqueness constraints involving start and end dates are removed (e.g. allowing someone to marry twice on the same date).

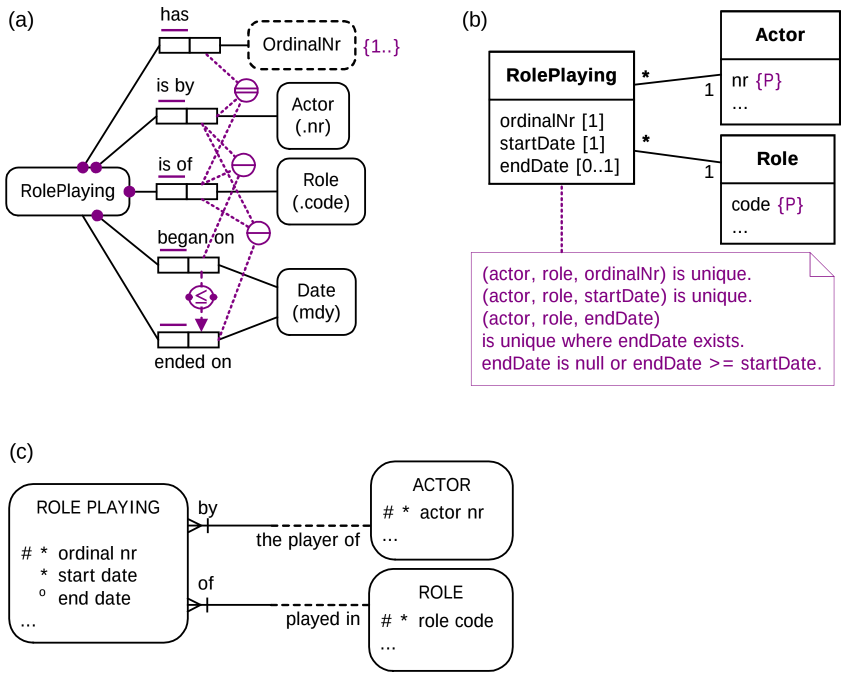


Figure 5. Another alternative repeatable role-playing pattern.

Conclusion

This article illustrated the use of the *repeatable role-playing pattern* to model history of objects as they move from one role type to another. This pattern is more general than the decreasing disjunctions pattern and the once-only role-playing pattern considered in previous articles, since those patterns are restricted to cases where the roles are once-only (once you've played and left that role, you never play it again). The repeatable role-playing pattern may be used not only for those cases, but also to maintain history in cases where a role may be repeatedly played more than once during an object's lifetime (e.g., 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).

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. Halpin, T. & Morgan, T. 2008, *Information Modeling and Relational Databases, 2nd edition*, Morgan Kaufmann, San Francisco.
5. 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.
6. 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.
7. Halpin T. 2007, 'Temporal Modeling (Part 1)', *Business Rules Journal*, Vol. 8, No. 2 (Feb. 2007), URL: http://www.BRCommunity.com/a2007/b332.html.
8. Halpin, T. 2007, 'Temporal Modeling (Part 2)', *Business Rules Journal*, Vol. 8, No. 6 (June 2007), URL: http://www.BRCommunity.com/a2007/b351.html.
9. Halpin, T. 2007, 'Temporal Modeling (Part 3)', *Business Rules Journal*, Vol. 8, No. 11 (Nov. 2007), URL: http://www.BRCommunity.com/a2007/b374.html.
10. Halpin, T. 2008, 'Temporal Modeling (Part 4)', *Business Rules Journal*, Vol. 9, No. 4 (Apr. 2008), URL: http://www.BRCommunity.com/a2008/b411.html.
11. Halpin, T. 2008, 'Temporal Modeling (Part 5)', *Business Rules Journal*, Vol. 9, No. 10 (Oct. 2008), URL: http://www.BRCommunity.com/a2008/b444.html.
12. Halpin, T. 2008, 'Temporal Modeling (Part 6)', *Business Rules Journal*, Vol. 9, No. 12 (Dec. 2008), URL: http://www.BRCommunity.com/a2008/b454.html.
13. Hay, D. (2006). *Data Model Patterns: A Metadata* Map. San Francisco: Morgan Kaufmann.
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.
16. Silverston, L. (2001). *The Data Model Resource Book: Revised Edition*, New York: Wiley.