Temporal Modeling: Part 2

Terry Halpin
Neumont University

This is the second in a series of articles on the impact of *time* on the conceptual modeling of business domains. The previous article [6] distinguished three basic temporal data types: *instant* (point in time), *interval* (duration of time), and *period* (anchored duration of time). It then classified *temporal object types* into *once-only* (e.g., Date) and *repeatable* (e.g., WeekDay) object types, It then 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). Finally, it showed how to model temporal details about point events or period events underlying instances of once-only fact types that are unchangeable.

The focus of this article is on modeling of temporal information about events underlying *changeable fact types*. Three graphical notations are used for examples: second generation Object-Role Modeling (ORM 2) [4, 5] as supported by the open source (Neumont ORM Architect) NORMA tool [3, 7]; the Unified Modeling Language (UML) [8, 9]; and the Barker notation [1] for Entity-Relationship Modeling (ER) [2].

Snapshots of Changeable Fact Types

Consider a medical clinic where patients are identified by patient numbers, have their birthdate recorded, and optionally have their weight, temperature, and allergies (if any) recorded. Figure 1 models this domain in ORM, UML, and Barker ER. As a non-standard extension to UML, we use the notation "{P}" to flag an attribute as a component of the preferred, natural identifier for instances of a class. The ORM and UML schemas include a derived fact type to compute a patient's age. A derivation rule may be added as a textual note for computing age from birthdate and current date (in ORM, the latter is obtained from the implicit, predefined fact type Date is today). For Barker ER, which has no notation for derived information, the derivation rule could be added in a separate document.

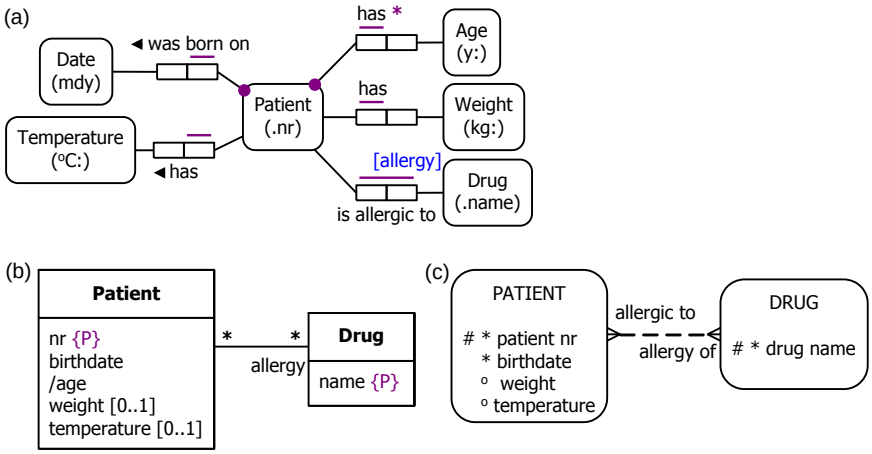

Figure 1. A simple patient schema in (a) ORM, (b) UML, and (c) Barker ER.

The fact type Patient was born on Date is a once-only fact type, and hence is *unchangeable* (ignoring the possibility of error correction, which is out of scope for this article). Each of the other fact types is *individually changeable* (i.e. its individual instances may change over time). The fact type Patient has Weight is *functional* for Patient (i.e. at any time, a patient has at most one weight), and is repeatable (patients may

gain or lose weight and later return to their previous weight). The temperature fact type is also functional and repeatable. The allergy fact type is *m:n* (many-to-many) and repeatable (patients may cease to be allergic to a given drug, but later become allergic to it again). The derived age fact type is functional and changeable even though it is not repeatable. Being functional, the age and weight fact types can be changeable only at the instance level—an old instance can change only by being replaced by a new instance. The non-functional allergy fact type is also *collectively changeable* (i.e. its collection or population of instances may change over time, by adding new instances or deleting old instances).

The individual changeability of the age, weight, temperature, and allergy fact types follows from their *time-deictic* nature. Deixis[1] (Greek for indicating, or point of reference) is the process whereby the meaning of a linguistic structure depends on its communication (utterance or inscription) context, and may be classified into five basic categories: time, place, person, discourse, and social. For example, the sentence "You are now reading this article" is person-deictic ("you"), time-deictic ("are", "now"), and discourse-deictic ("this article").

In time (or temporal) deixis, the meaning of a communication act depends on the time in which it was performed. Temporal deictic reference may appear via temporal adverbs (e.g., "now", "currently", "today", "tomorrow", "yesterday"), *tense* (e.g. "is eating", "was eating", "will eat"), combinations of deictic modifiers and time units (e.g. "last year", "this week", "next month"), or other temporal phrases (e.g., "two days ago", "within a month"). In information modeling, time-deictic aspects of predicates are usually restricted to tense and temporal adverbs.

In Figure 1(a), the birthdate predicate is in past tense (was born on), while the other four predicates are in present tense (has, has, has, is allergic to). Since one cannot change the past, instances of the birthdate fact type are unchangeable. If we wish our information model to remember instances of birthdate facts, these facts also cannot be deleted. In contrast, instances of the four present tense fact types are changeable, and may be replaced by other instances. Consider the following fact instance:

Patient 101 has Weight 80 kg.

This is shorthand for "Patient 101 *currently* has Weight 80 kg", which itself is shorthand for "Patient 101 has Weight 80 kg at time *t*, where *t* is the time at which the sentence was communicated". Originally *t* is a valid time (e.g. the time at which the patient's weight was actually measured). Typically, the fact is then entered into the information model at some later transaction time, and remains there until it either is updated by a more recent measurement or is deleted. From the viewpoint of the information model, the fact really means "*When last inspected*, Patient 101 had Weight 80 kg". For such reasons, the fact is known as a *snapshot* fact. In practice, if the delay between valid and transaction times is minor, and the property (e.g. weight, allergy) is re-evaluated on a suitably regular basis (which may be very infrequent if the property changes only slowly over time), then business users are willing to treat snapshot facts as if they were current.

Maintaining History of Functional, Changeable Fact Types

If a fact type is individually changeable, and we need to maintain a *history* (full or partial) of its instances, then we should add or adapt a fact type to achieve this, using a temporal object type of the desired temporal granularity. In ORM and UML, one simple way to do this is to insert into a "key" of the original, snapshot fact type a role that is played by the temporal object type, rephrasing the predicate as appropriate. A key of a fact type is a set of its roles that is exactly spanned by a uniqueness constraint. For example, in Figure 1(a) the key of the weight fact type is the role played there by Patient. Suppose we wish to maintain a history of patient weights where for each patient at most one weight measurement is performed per day. Since our temporal granularity is one day, we choose Date as the object type and insert its relevant role into the binary fact type to produce the ternary fact type Patient on Date had Weight, and expand the key uniqueness constraint to cover this role as shown in Figure 2(a). The uniqueness constraint verbalizes as: **For each** Patient **and** Date, **that** Patient on **that** Date had **at most one** Weight. The UML version is shown in Figure 2(b).

For Barker ER, which does not support ternary fact types, a binary version is formed by introducing the entity type WeightMeasurement, which is identified by combining the date attribute with the patient whose weight is measured, as shown in Figure 2(c). In Barker ER, an octothorpe "#" and a stroke "|" respectively indicate attributes and relationship roles that are primary identifier components.

[1] http://en.wikipedia.org/wiki/Deixis

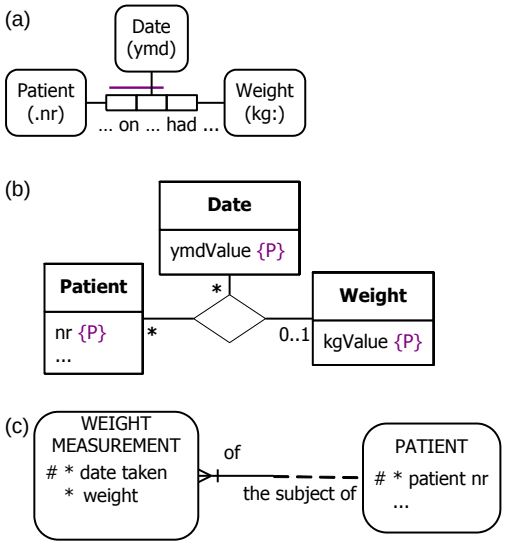

Figure 2. Maintaining history of patient weights.

As an alternative, both ORM and UML (but not Barker ER) allow associations to be objectified, as shown in Figure 3.

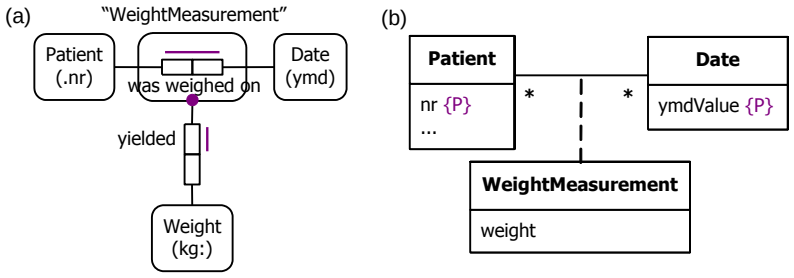

Figure 3. Objectifying the weight measurement association in (a) ORM and (b) UML.

As another alternative, both ORM and UML allow WeightMeasurement to be introduced as in the earlier Barker ER solution, without needing to objectify an association, as shown in Figure 4. The ORM schema uses a preferred external uniqueness constraint (circled double bar) to indicate that WeightMeasurement is identified by combining the date and patient. As UML has no graphic constraint for external uniqueness, this constraint is declared here informally in a note, as shown in Figure 4(b). If desired it could also be declared formally in OCL.

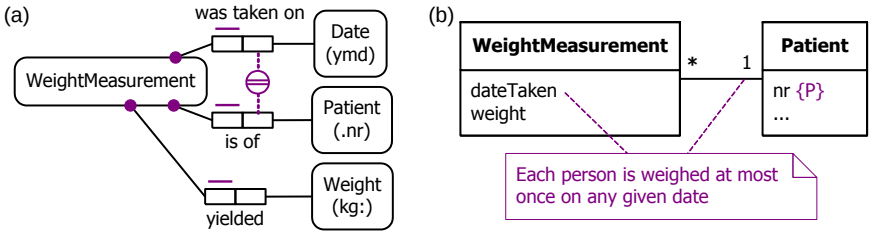

Figure 4. Modeling WeightMeasurement without objectification in (a) ORM and (b) UML.

UML assumes that each object is identified by an internal surrogate. If desired, we could add a simple, visible identifier (e.g., a measurement number) for WeightMeasurement. This could be done for ORM, UML, and Barker ER, as shown in Figure 5. The combination of date and patient now provides a secondary identifier for WeightMeasurement, as shown by ORM's simple external uniqueness constraint (circled single bar), and by the UML note. The Barker ER notation used earlier to capture this constraint can no longer be used, since it applies only to primary identifiers; so this constraint should be noted somewhere in a separate document.

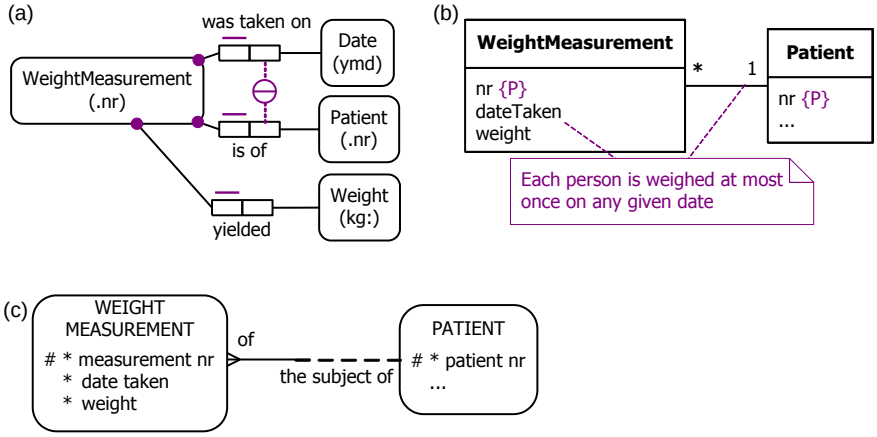

Figure 5. Adding a simple, visible identifier for WeightMeasurement in (a) ORM, (b) UML, and (c) Barker ER.

Conceptually, capturing history by expanding the key with a temporal object type as in Figure 2(a) and Figure 2(b) is the simplest and least disruptive approach. However, introducing an object type such as WeightMeasurement either directly or by objectification is preferable if we now or later wish to record information about it (e.g. who performed the measurement).

The approaches just discussed are appropriate for maintaining history of patient temperatures. We might of course choose a different temporal granularity. For example, if our granularity is one hour, we could use the fact type Patient at Hour had Temperature.

Note that while the above approaches allow full history (at the desired granularity), they do not demand full history. For example, we are not constrained to record a patient's weight on a daily basis. Note also that given date arithmetic, there is no need to add any further fact types to Figure 1 to determine a history of how patients have aged over time (this is all derivable).

For comparison purposes, we sometimes require a well-defined partial history (e.g., current and previous values only). The basics for such a situation can be easily modeled by using multiple fact types, as shown for patient weights in Figure 6. A complete approach requires additional textual rules to assign the old current weight to the new previous weight on update. A high level language for specifying such rules will be discussed in a later article.

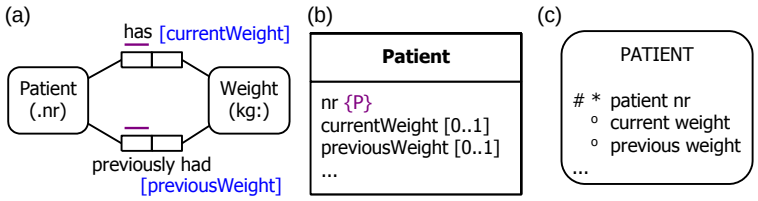

Figure 6. Recording current and previous weights in (a) ORM, (b) UML, and (c) Barker ER.

This article discussed how to maintain history of functional, changeable fact types like those in Figure 1. While these cases were straightforward, they helped to illustrate two modeling heuristics: (1) take care to use the appropriate tense when naming predicates and attributes; (2) for each changeable fact type, consciously decide what if any history needs to be maintained.

Maintaining history of non-functional fact types like the allergy fact type in Figure 1 is a little more complicated, since they may be both collectively and individually changeable. Such cases are considered in the next article, along with more complex cases involving open and closed periods.

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. 2001, *Information Modeling and Relational Databases*, 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. 2007, 'Temporal Modeling (Part 1)', *Business Rules Journal*, Vol. 8, No. 2 (Feb. 2007), URL: http://www.BRCommunity.com/a2007/b332.html.
7. NORMA website: http://sourceforge.net/projects/orm.
8. Object Management Group 2003, *UML 2.0 Infrastructure*, URL: http://www.omg.org/uml.
9. Object Management Group 2006, *Semantics of Business Vocabulary and Rules Interim Specification*. URL: www.omg.org/cgi-bin/doc?dtc/06-03-02.