PART I

OBJECT MODELING

In this section we provide a number of articles that define and refine issues and techniques associated with modeling objects. This is a dynamic and changing discipline. As you will see, the following contributors are among those driving the science.

Association Multiplicity

David M. Papurt

Relating a pair of abstract data types, association is a rich concept with many facets. My earlier columns[1,2] gave several descriptions of association: a referenced, named abstract data type; a bidirectional mapping between associated type extensions; a type with related extension and link instances; and an abstract data type with attributes, associations, and operations of its own. Each of these descriptions is from a distinct perspective and emphasizes an alternate aspect of the relationship; the latter descriptions unify seemingly different elements of the object model.

For simplicity, so far, this series of columns has restricted *association multiplicity* (or simply *multiplicity*); on the instance level, an association traversal in either direction has yielded exactly one instance. But in a more general and applicable model, a traversal can yield numbers of instances other than one. This column continues examination of association and focuses on associations wherein traversal can yield numbers of instances other than one.

First, mappings that yield multiple values are covered. Then, one-to-many and many-to-many associations are examined and described in terms of mappings that yield multiple values. Finally, association multiplicities other than one and many are examined.

Multiple Value Mappings

Mapping is a mathematical concept. My previous column[2] described two mappings that yield single values: unidirectional one-to-one mapping and bidirectional one-to-one mapping. This section extends that discussion and

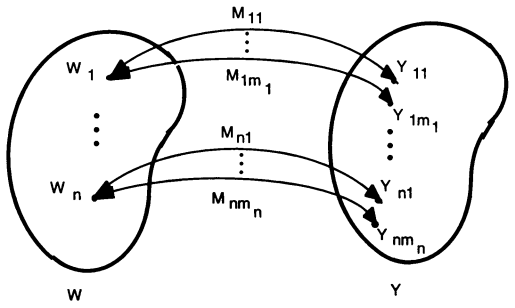

FIGURE 1A. *Bidirectional one-to-many mapping M. Sets W and Y alternate as range and domain.*

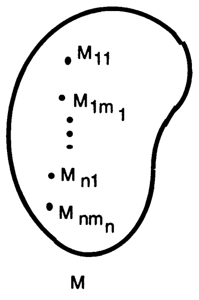

FIGURE 1B. *Mapping M viewed as a set.*

covers two additional mappings that yield multiple values: bidirectional one-to-many mapping and bidirectional many-to-many mapping.

Bidirectional One-to-Many Mapping

Figure 1 represents, with Venn diagrams, a bidirectional one-to-many mapping M two different ways. Figure 1A emphasizes that the mapping is a pair of inverse functions defined on two sets; W and Y alternate as domain and range, depending on mapping direction. Figure 1B emphasizes that the individual mapping instances form a set $\{M^{11}, \ldots, M^{1}m^{1}, \ldots, Mn^{1}, \ldots, Mnmn\}$.

The forward mapping function F: W→Y yields multiple values—i.e., is set valued. The reverse mapping function R: Y→W applied to any one of those multiple values yields the single value the forward function mapped originally. That is, F: W→Y and R: Y→W are constrained and must satisfy

```
R( OneOf( F(Wj) ) ) = Wj  j = 1, 2, . . ., n
where OneOf() selects any member of its set-valued argument.
```

Bidirectional Many-to-Many Mapping

Figure 2 represents, with Venn diagrams, a bidirectional many-to-many mapping N two different ways. Figure 2A emphasizes the mapping is a pair of inverse functions defined on two sets; D and V alternate as domain and range, depending on mapping direction. Figure 2B emphasizes that the individual mapping instances form a set $\{ND^{1}V^{1}, \ldots, NDdVv\}$.

Both the forward mapping function F: D→V and reverse mapping function R: V→D yield multiple values. The reverse mapping function R applied to any one value generated by the forward mapping function F yields a set

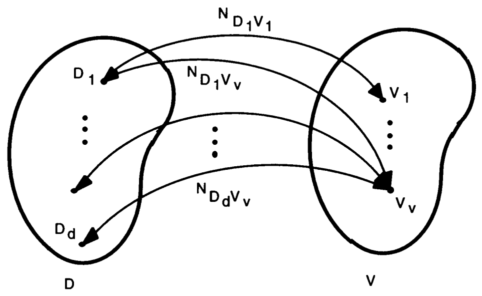

FIGURE 2A. *Bidirectional many-to-many mapping N. Sets D and V alternate as range and domain.*

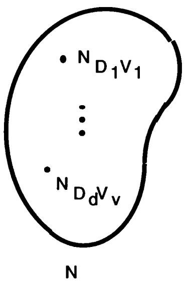

FIGURE 2B. *Mapping N viewed as a set.*

that includes as a member the value the forward function mapped originally. That is, F: D→V and R: V→D are constrained and must satisfy

```
Dj ∈ R( OneOf( F(Dj) ) )
     j = 1, 2, . . ., d
```

where ∈ means *is an element of.*

ONE-TO-MANY MULTIPLICITY ASSOCIATION

Association multiplicity is the possible numbers of instances of one type participating in the association that can link to a *single* instance of the other type participating in the association. Generally, the possible numbers of instances, i.e., the multiplicity, is expressed as a set of non-negative integers. It follows from the intrinsic bidirectionality of association that there are two multiplicities for each association, one for each direction of traversal. Object type diagrams represent multiplicity with special symbols for commonly occurring multiplicities and explicit lists of numbers and intervals for less common multiplicities.[3]

TIP

Multiplicity is expressed as a set of non-negative integers.

The `Automobile-Person` Ownership association described in my previous column[2] had one-to-one multiplicity; the notation was a solid line between abstract data types. More precisely, in that model, every single instance of `Automobile` must link with exactly one `Person` instance, and every single instance of `Person` must link with exactly one `Automobile` instance.

Figure 3 represents the more complex `Automobile-Person` Carries (as

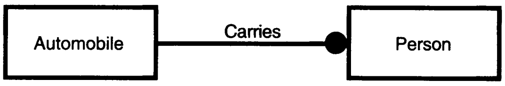

FIGURE 3A. *Automobile-Person Carries association. Object type diagram.*

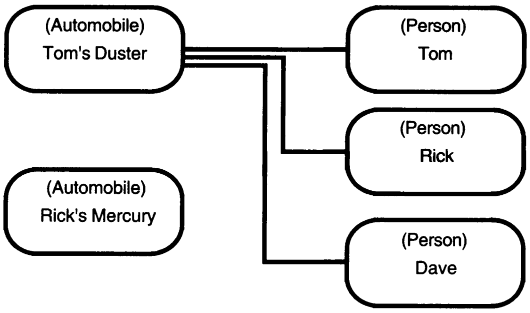

FIGURE 3B. *Sample Automobile and Person instances and Carries association links. Object instance diagram.*

passenger) association. The association is loosely described as *one-to-many,* but the solid-ball symbol in the object-type diagram of Figure 3A has more precise meaning. Reflecting that an `Automobile` can carry multiple `Persons`, the notation means that any non–negative integer (i.e., 0, 1, 2, . . .) number of `Person` instances can link to a single instance of `Automobile`. (For simplicity, this example disregards the practical maximum number of passengers that can fit into an `Automobile`.) Reflecting that a `Person` rides in only one `Automobile` at a time, the solid line going into `Automobile` means that exactly one Automobile must link to each single instance of `Person`.

The object instance diagram in Figure 3B represents a particular instance constellation—one `Automobile` carrying three `Person` instances and another empty of any `Persons`. In compliance with the association multiplicity constraints, each `Automobile` links with any number of `Persons`, and each `Person` links with exactly one `Automobile`.

One-to-Many Association as Mapping and Type

A one-to-many association is a bidirectional one-to-many mapping between type extensions. As for one-to-one association,[2] the mapping domain and range are the extensions of the types participating in the association, and they alternate depending on mapping direction.

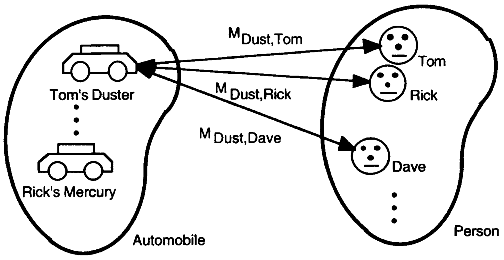

FIGURE 4A. *Automobile–Person Carries association viewed as mapping.*

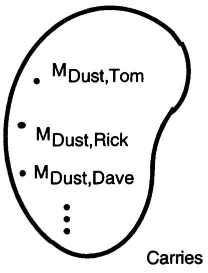

FIGURE 4B. *Mapping viewed as a set, the extension of type Carries Association.*

The mapping instances—i.e., links—form a set, and the set is a type extension. Thus, a one-to-many association is a type with instances that are conceptual or actual links between associated abstract data type instances.

Figure 4 represents the `Carries` (as passenger) one-to-many association between abstract data types `Automobile` and `Person`. This Figure expresses information equivalent to Figure 3 but with different emphasis. Figure 4a emphasizes that the `Carries` association is a bidirectional one-to-many mapping, and Figure 4B emphasizes that the association mapping instances— links—form a set; that set is the extension of type `Carries Association` between `Automobile` and `Person`.

Further Observations

Unlike a one-to-one association,[2] a one-to-many association is unbalanced. The imbalance corresponds to the differing multiplicities at either end of the association. But, as in the one-to-one case, there is no preferred direction of the association, and the association can be traversed in either direction.

As in the earlier one-to-one case,[2] a link of a one-to-many association is not part of either of the two instances it connects. The link depends on both instances and contains information transcending the information in the instances.

When implementing in a programming language, often, for simplicity, a one-to-many association is implemented in a single direction only; the association can be implemented so that the link is traversed directly from the one side to the many side, or so the link is traversed directly from the many side to the one side. But never confuse such implementation issues with the conceptual model.

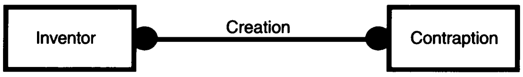

FIGURE 5A. *Inventor-Contraption Creation association. Object type diagram.*

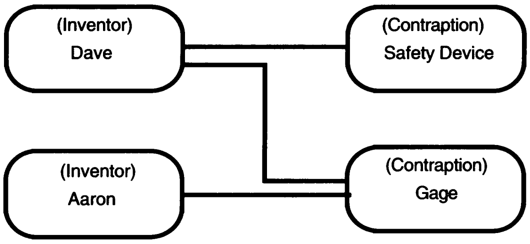

FIGURE 5B. *Sample Inventor and Contraption instances and Creation association links. Object instance diagram.*

MANY-TO-MANY MULTIPLICITY ASSOCIATION

Figure 5a represents the *many-to-many* Creation association between abstract data types `Inventor` and `Contraption`. An `Inventor` can (and usually does) create multiple `Contraptions`, and multiple `Inventors` can work together to create one `Contraption`; the solid balls indicate that any non-negative integer number of `Contraption` instances can link to a single instance of `Inventor`, and any non-negative integer number of `Inventor` instances can link to a single instance of `Contraption`.

The object instance diagram in Figure 5B represents a particular instance constellation—two `Inventors` and two `Contraptions` connected by three links. In compliance with the association multiplicity constraints, each `Inventor` links with any number of `Contraptions`, and each `Contraption` links with any number of `Inventors`.

TIP

A many-to-many association describes conceptual or actual connections.

Like a one-to-one association and one-to-many association—and like associations of any multiplicity—a many-to-many association is a bidirectional mapping with domain and range that are the extensions of the types participating in the association; in this case, the mapping is many-to-many. Further—and, again, like associations of any multiplicity—a many-to-many association is a type describing conceptual or actual connections between abstract data type instances. Like any type, related instances

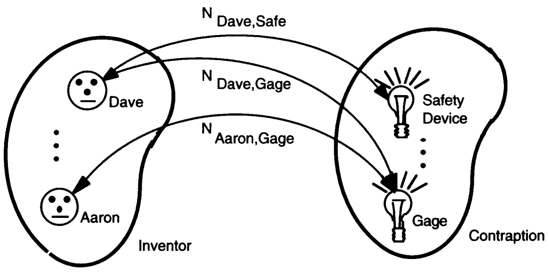

Figure 6a. *Inventor–Contraption Creation association viewed as mapping.*

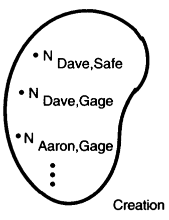

Figure 6b. *Mapping viewed as a set, the extension of type Creation Association.*

and an extension of all instances exist; the instances are links, and the extension is the set of all links of the association. Figure 6 represents the many-to-many Creation association between Inventor and Contraption as a mapping and as a type.

Other Multiplicities

An association multiplicity constrains the number of instances of one type participating in an association that can link to a single instance of the other type participating in the association. In general, a multiplicity is a set of non-negative integers; the actual number of instances of one type linking to a single instance of the other type can be an integer only from this set. There are two multiplicities for each association, one for each direction of traversal.

Object-type diagrams denote multiplicities with special symbols for common multiplicities and explicit lists of numbers and intervals for less common multiplicities. Earlier sections presented the solid line without any terminator—indicating one—and the solid ball—designating any non-negative number (many). A hollow ball indicates optional—zero or one only—multiplicity.

Explicit lists of numbers and intervals written next to the association line near a type rectangle designate less common multiplicities without special symbols. For example, 1+ indicates any number greater than zero, 2–6 designates two to six inclusive, and 3, 5, 8–10 means three, five, eight, nine, or ten.

TIP

This section describes the nomenclature.

The object type diagram in Figure 7A uses an explicit list of numbers to

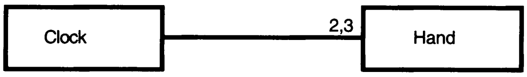

FIGURE 7A. *Clock-Hand association. Object type diagram.*

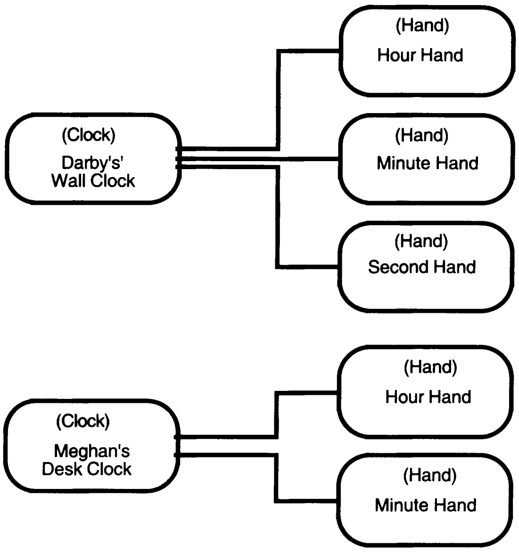

FIGURE 7B. *Sample Clock and Hand instances and association links. Object instance diagram.*

indicate that every `Clock` can link with either two or three Hands—any more or less violates the multiplicity constraint—and every `Clock Hand` links with a single `Clock`. The object instance diagram of Figure 7B shows a particular instance constellation.

The object type diagram in Figure 8A represents the relationship between types `Street`, `Intersection`, and `Traffic Light` and includes two associations; the multiplicity constraints are more complex than any seen to this point. A `Street` can link with as few as zero `Intersections`. But the 2_+ notation indicates that each `Intersection` must link with at least two `Streets`; an `Intersection` simply cannot exist unless at least two `Streets` meet or cross. An `Intersection` optionally links to a `Traffic Light`, as indicated by the hollow-ball—zero or one—multiplicity symbol, and every `Traffic Light` links to exactly one `Intersection`.

The street map of Figure 8B and object instance diagram of Figure 8C rep-