ROAD981.TEX

Clarifying specialized forms of association in UML and OML

D.G. Firesmith and B. Henderson-Sellers

for ROAD/JOOP May 1998, Vol 11 no 2

Deadline 17 February

In my last two columns, I have been discussing some of the details of association and aggregation relationships in OPEN's preferred modelling language, OML (OPEN Modelling Language) (ref. 1) as well as drawing parallels between the support for various relationships offered in both OML and UML (Unified Modeling Language: refs. 2,3). In this month's column, I am joined by my colleague in the OPEN Consortium, Don Firesmith, in a more detailed discussion of the support for specialized forms of association, particularly aggregation, at not only the analysis level (as discussed in ref. 4) but also in detailed design and implementation/coding.

WHAT IS THE PROBLEM?

Aggregation is primarily a modelling tool. In order to assist software developers understand, model and represent their application, most OO methods have identified some form of aggregation (or whole-part relationship) as being useful in many situations. One problem is that the various types of aggregation discussed in refs. 4–6 (Figure 1) are not supported explicitly at the coding level. Furthermore, when discussing aggregation with any programming language expert, the concerns are not those of the configurational (where the parts are structurally or functionally related) or the homeomerous (where the parts are of the same substance as the whole) nature of the relationship (Figure 1) but include those of lifetime dependency, coincident destruction and whether to use a reference, embedded value or pointer.

WHOLE–PART AS A CONCEPTUAL RELATIONSHIP

The meaning of all whole-part relationships requires that the whole is more than the parts (i.e. there is an emergent property). Figure 1 identifies three modelling-level characteristics of whole-part (or meronymic) relationships (ref. 5): configurational, home-omerous and invariance — although there are likely to be others which might be useful in discriminating between Material–Object and Place–Area for instance. It is recommended (ref. 4) that the four configurational relationships in Figure 1 be grouped together as a single relationship, called simply aggregation in OML, whilst the three non-configurational whole-part relationships be grouped as a new OO relationship named Membership. The difference between these two whole-part relationships is further clarified in Figure 2. In part (a) is seen an aggregational (i.e. configurational) whole-part relationship. The notion of being configurational requires that the parts bear some structural or functional (i.e. configurational) relationship to each other — in other words, there must be a relationship(s) between two or more parts. For example, a car engine is an aggregate of its parts — the way the parts are configured together is critical to the existence of a functioning engine. This is shown here by an association between the parts themselves. It should be noted that configurational relationships include both variant and invariant styles (Figure 1). In OML, these are grouped together as the basic OML definition of aggregation; in UML a composite or strong aggregation permits change of ownership thus suggesting that it represents only the configurational, variance element of Figure 1 (called there Component–integral object). Part (b) of Figure 2 illustrates the second OML whole-part relationship of Membership in which the parts do NOT bear any functional or structural relationship to each other (there is no necessary connecting association relationship between any of the parts — any such connexion being incidental). Membership is not discussed in UML — we presume that it would have to be modelled as an association.

AGGREGATION IN DETAILED DESIGN

In UML, on the other hand, the explicit focus is not on whether an aggregation is or is not configurational (this is implicit as discussed above) but on whether parts are shared and whether lifetimes are coincident. The black and white diamonds in UML's notation address, at the same time, both (a) the idea of shared/unique ownership and (b) the notion that when the aggregate is destroyed the part may not be destroyed if it is shared (white diamond) and must be destroyed if ownership if unique (black diamond) (refs. 2,7). Arguments rage in the OTUG newsgroup about not only this but also whether the black and white diamonds imply by-value and by-reference, much as in the original Booch notation (although there is no discussion of this topic in ref. 2). Arguments on this and other newsgroups leads us to realize that in fact we are talking about three separate ideas/concepts (by-value/by-reference; sharing; lifetime dependency) which are orthogonal to each other and also to the configurational arguments above. UML may confound the first three because they may be statistically correlated: (i) by-value with not sharing with lifetime dependent and (ii) by-reference with sharing with lifetime independent being frequently encountered. This is particularly true at the implementation level.

Certainly the need in detailed design to annotate diagrams to indicate by-value or by-reference as well as an indication of lifetime coincidence cannot be denied. However, it is only part of the answer to modelling aggregations and, in the use of black and white diamonds in UML (which require rote memorization in any case), there is no clear mapping between these notational symbols and these two orthogonal concepts (by-value/by-reference and lifetime dependencies) as well as configurational issues.

Furthermore, the notion of sharing is *not* a property of an aggregation (or indeed any other) *relationship*. Sharing states that an object (e.g. as a part in an aggregation) is referenced by two aggregates (Figure 3) — in other words, there must exist at least two such relationships (two white diamonds in UML). Thus, although Figure 4 is helpful in pointing out that the published aggregations in OML are in fact not conflicting but orthogonal with those in the published UML, and that this is clearly shown by the use of discriminators, it remains misleading in the sense that the main discriminant of "sharing" obfuscates (at least) two important issues and introduces another issue (sharing) which is not a characteristic of a relationship but of the class itself. Furthermore, sharing (multiple references to one server from several clients) occurs for *all* Referential Relationships, not just aggregation — although in UML it is only applied in this one special case.

CONTAINMENT CONFUSED WITH THE WHOLE–PART RELATIONSHIP

Containment is often confused with aggregation. However, it is NOT a whole-part relationship (refs. 4–6). In a containment relationship, the contents of the container have an existence independent of their location within the container. Furthermore, a container can be empty whereas in a whole-part relationship, the whole cannot exist at all if there are no parts. A further difference is that the container class does not know about its contents and can thus only send the most general object-level messages to its contents whereas it is common for the aggregate object in an aggregation to delegate messages very specifically to its parts.

AN OVERALL SOLUTION

The overall solution is to discriminate clearly between a number of orthogonal concepts. The first relates to Figure 1 and identifies whether a whole-part relationship is configurational or not. It may also be useful to identify, in some very special cases, whether the configurational aggregation is or is not invariant and this can be accomplished in either UML or OML notation by use of stereotypes — although the black diamond default in UML seems to reflect a variable configurational whole–part relationship (ref. 2, page 38). (The homeomerous characteristic seems to us to have significantly less value in real-life modelling situations). In fact, this is an ideal use of stereotypes which should identify metaclass specializations in the form of partitions. This tends to have an analysis-level focus.

Figure 5 summarizes the relationships (in the OML metamodel) between AGGREGATION and MEMBERSHIP (both being whole-part relationships) together with CONTAINMENT and the third subtype of REFERENTIAL RELATIONSHIP, the NORMAL ASSOCIATION (see also ref. 8). The OML notation (lower part of figure) for all referential relationships is a directed arrow. For relationships other than the normal association, an annotation is added at the "parent" end. For containment, a stylized cup icon is used inside a circle; for aggregation a plus sign to show that the aggregate is (more than) the sum of its parts; for membership, an epsilon which is the set membership symbol. In addition, as noted above, a group must contain at least one member to remain in existence whilst a container may exist in an empty state. Since there is an additional configuration constraint between the parts of an aggregate, there must exist at least two parts. (All these constraints are shown by using multiplicities as depicted in Figure 5).

In addition, we need to include a notation (and underpinning metamodel fragment) to clearly differentiate between the concepts of (at least) by-value/by-reference and lifetime dependencies — as discussed above, sharing is not part of aggregation modelling and needs no specific notation. We thus recommend (a) a notation (black tombstone icon at part end) to indicate whether or not the parts must die at the same time as the whole (independent versus non-independent) together with (b) a stereotype to clarify by-value versus by-reference rather than permitting users to insinuate this meaning into the meta-model/notation. That these are both necessary is easily seen. A by-value implementation can be regarded as one in which the part is fully encapsulated within the "parent". Whilst this would be implemented directly in C++, this is not possible in Java and Smalltalk and, in fact, all such relationships would have to be by-reference. Conversely, if a by-reference implementation is used, this could be from one whole to one part (when it is likely that destroying the whole will destroy the part), or two wholes to one part (sharing: when it is most unlikely that destroying the whole will necessarily destroy the part) to a membership in which there is one whole and several parts (e.g. a fleet is made up of ships) and destroying the whole is most unlikely to destroy the parts (although of course it could). For instance, one could scatter (and thus destroy) a fleet but retain all ships as individuals or one could destroy a fleet in a war by physically destroying all the ships, perhaps individually or by a single explosion. It is thus clear that attempting to link lifetime coincidence with by-value or by-reference or to sharing/unique ownership is a fruitless exercise.

As well as these new partitions, there are several others which can be identified at the level of REFERENTIAL RELATIONSHIP, and are thus inherited not only by normal associations but also by containment and both whole-part relationships (as illustrated in Figure 5). The most relevant of these to the discussion here are variable versus constant (to reflect the Invariance column in Figure 1) and used versus not-used (Figure 6), the latter describing whether the message path determined by the relationship arrow will support "traffic" (i.e. messages) along it ("used") or whether it remains a static, architectural linkage ("not-used").

Multiplicity can be used to indicate the mandatory/optional partition of Figure 6, an annotation (tombstone icon) for dependence/independence and the remainder of the partitions are best depicted as stereotypes (an example is shown in Figure 7).

In OML Version 1.1 (which we discuss fully in our next two columns), we thus add to Version 1.0 (ref. 1) for aggregation relationships:

- the notation for membership (refs. 4,8) as shown in Figure 5
- a new notation for by-value or by-reference which is simply an optional stereotype annotation to the relationship arc (as shown in Figure 7). The options here are simply {by-value} and {by-reference}.
- a new notation for the case when the lifetime of the part is necessarily terminated subsequent to the destruction of the aggregate itself. The notation here is consistent with the rest of the COMN/OML notation and is a black tombstone icon placed at the arrowed end (part end) of the aggregation relationship (Figure 7). The tombstone is visually indicative of death. The annotation is useful for assisting in programming. If the death of the parts is either not contingent on the destruction of the whole or if this is not known, then the tombstone is not shown.
- stereotype labels to indicate used/not used and variable/constant (Figure 7).
- multiplicity to also include optional or mandatory [zero or one at lower bound].

With these orthogonal sets of notation, it is now possible to build up a true picture of possible combinations.

CONCLUSIONS

The additional notation proposed here for OML (and potentially for a future version of UML) permits clarification of whole-part characteristics and extends the notational usage across the lifecycle since parts of this recommended notation pertain more to modelling (analysis and early design) and parts pertain solely to programming. It separates concerns at the detailed design/implementation level that have caused excessive discussion in the newsgroups and which are not clearly explained in the UML documentation (refs. 2,3).

Acknowledgements

This is contribution number 98/1 of the Centre for Object Technology Applications and Research.

References

1. Firesmith, D.G., Henderson-Sellers, B. and Graham, I., 1997, *OPEN Modeling Language (OML) Reference Manual*, SIGS Books, New York, USA, 271pp

2. OMG, 1997, UML Semantics. Version 1.1, 15 September 1997, OMG document ad/97–08-04

3. OMG 1997, UML Notation. Version 1.1, 15 September 1997, OMG document ad/97–08–05

4. Henderson-Sellers, B., 1997, OPEN relationships — composition and containment, *J. Obj.-Oriented Prog.,* **10(7)**, 51–55

5. Winston, M.E., Chaffin, R. and Herrmann, D., 1987, A taxonomy of part–whole relations, *Cognitive Science,* **11**, 417–444

6. Odell, J.J., 1994, Six different kinds of composition, *J. Obj.-Oriented Prog.,* **6(8)**, 10–15

7. Fowler, M. and Scott, K., 1997, *UML Distilled. Applying the standard object modeling language*, Addison-Wesley, Reading, MA, 179pp

8. Henderson-Sellers, B., 1998, OPEN relationships — associations, mappings, dependencies, and uses, *JOOP*, **10(9)**, 49–57

Figure legends

Figure 1 Seven types of meronymic (aggregation) relationship proposed in ref. 4.

Figure 2 (a) Aggregation is a configurational relationship and there are thus connexions between the parts; (b) Membership is a non-configurational relationship with no prerequisite for any such connectivity at the part level.

Figure 3 Illustration of the use of the UML white diamond for sharing. Sharing is a construct in which one server must be referenced by at least two clients.

Figure 4 Metamodel for whole-part relationships showing two discriminants: one for configuration and one for sharing.

Figure 5 Metamodel hierarchy fragment for REFERENTIAL RELATIONSHIP and its subtypes together with OML notation for Association, Aggregation, Membership and Containment.

Figure 6 Metamodel fragment for REFERENTIAL RELATIONSHIP showing five of its partitions.

Figure 7 Example of a whole-part, referential relationship showing the use of stereotypes and tombstone annotation.