The Baseless Links Problem

Gonzalo Génova, Juan Llorens, José M. Fuentes
Computer Science Department, Carlos III University of Madrid
Avda. Universidad 30, 28911 Leganés (Madrid), Spain
{ggenova, llorens}@inf.uc3m.es, jmiguel@ie.inf.uc3m.es

Abstract

The UML Standard does not consistently follow the rule that every link is an instance of an association, leading to consistency problems in the metamodel itself, as well as in user models. The proposed solution reaffirms this rule, implying a slight change to the metamodel, and gives guidelines for CASE tools.

1. Introduction

One of the rules that are expected to be fulfilled in a well-formed UML model is that every instance has a type (in some cases, an instance can have more that one type). If this rule is not satisfied, we can say that there exists an instra-consistency problem within the model.

A link between objects is a special kind of instance, namely, an instance of an association between classifiers. However, this principle is not clearly and consistently followed in the UML Standard [2][1]. As we are going to see, there are common modeling situations in which, apparently, there are links that are not instances of associations (thus, *baseless links*).

2. Is every communication link an instance of an association?

Consider the example in Figure 1(a): there is a class diagram with a one-way association from class `Owner` to class `Bank`, and another one-way association from class `Owner` to class `Account`. In Figure 1(b) there is a collaboration diagram where an object of class `Owner` sends a message to an object of class `Bank` containing an object of class `Account` as argument, and the `Bank` object uses this `Account` object to send it a message: the owner object communicates its bank to close its account. In UML this interaction is usually modeled using a stereotyped «`parameter`» link from the `Bank` object to the `Account` object. Is this a true link, or is it only a graphical fiction? Does this link require an association between the `Bank` and `Account` classes? If not, does this mean that the link is not an instance of any association? This question is far from having being clarified, as recent research demonstrates[3][2].

The Standard is rather contradictory in this respect, since it gives two different solutions to thisproblem:

- *Sometimes a message does not use a communication link*. After stating that a message instance (a.k.a. stimulus) "uses a link between the sender and the receiver for communication", the Standard acknowledges some *special situations* in which this communication link may be missing: "if the receiver is an argument inside the current activation, a local or global variable, or if the stimulus is sent to the sender instance itself" [UML, p. 2-114]. Therefore, the link `myBank→myAccount` in Figure 1(b) would be a *fiction*, and no association is required between `Bank` and `Account`.
- *Sometimes a link is not an instance of an association*. The Standard defines *five standard stereotypes* for `LinkEnd` («global», «local», «parameter», and «self», in addition to the redundant «association») to handle those same special situations [UML, p. 2-103], where we find communication without associations. Therefore, the link `myBank→myAccount` in Figure 1(b) would be a *true link*, but a link that is not derived from the existence of an association between `Bank` and `Account`, but from "other circumstances". Again, no association is required between `Bank` and `Account`.

[1] We quote, by section and page numbers as usual, version 1.4 of the UML Standard. Thus, "[UML, p. 2-114]" means "[2] sect. 2, p. 114". Version 1.5 has not introduced any changes regarding the subject of this paper, and version 2.0 is not yet approved and available to the general public.

[2] See also the contributions to The Precise UML Group mailing list [4] during the years 2000-2001 under the subjects "Links & messages", "Link as instance, tuple, path", "Sets and bags", and "Dependencies and associations", where the authors played an active role. Other similar problems involving compound navigation expressions and stereotyped «`self`» links are described in a recent paper by the authors [1]. We are not aware of any proposal to UML 2.0 that addresses this issue.

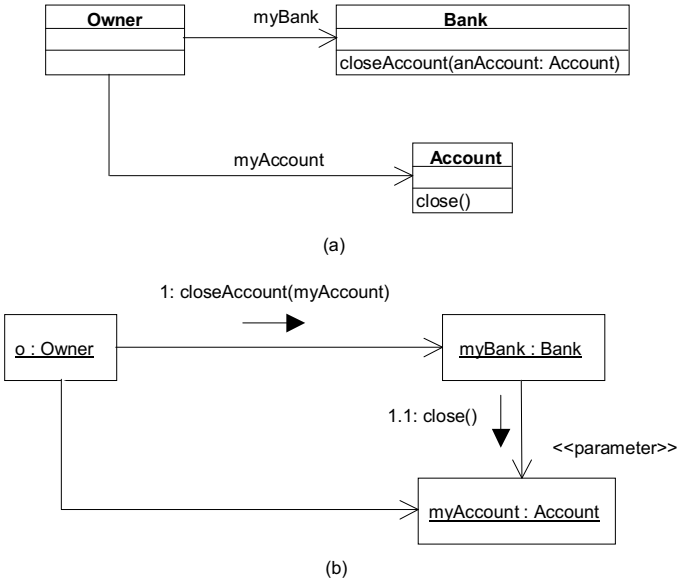

Figure 1. (a) Class diagram with classes `Owner`, `Bank` and `Account`, and two associations among them. (b) Collaboration diagram using a stereotyped «`parameter`» link without any existing association `Bank`→`Account`

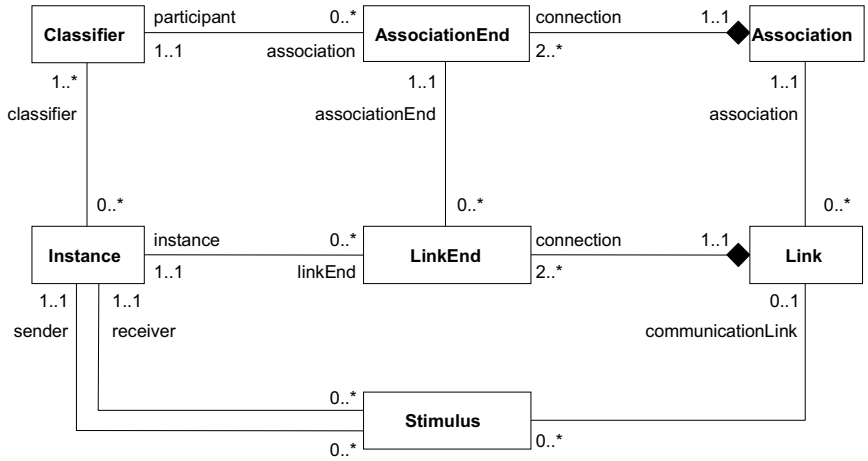

Figure 2. Metamodel of communication links extracted from Figures 2-6, 2-16 and 2-17 in the Standard

The first solution is consistent with the statement that a link *is* an instance of an association, represented in the metamodel by a mandatory `Association` that specifies the `Link` (multiplicity 1..1 on the role `Link.association`, see Figure 2), and it is also consistent with the statement that the link is *optionally* used by a message for communication (multiplicity 0..1 on the role `Stimulus.communicationLink`): sometimes the message uses a link (which is an instance of an association), and sometimes the message does not use any link (so no association involved).

The second solution (stereotyped links) breaks the principle that every link is an instance of an association, and it contradicts the first solution (optional links): if the communication link is optional, what is the sense of defining these special stereotyped links? Nevertheless, this solution is consistent with the common representation of interactions in collaboration diagrams, where a message *always* uses a link.

None of these two solutions is satisfactory. If links are optional, what is the representation of a message that is sent through a missing link in a collaboration diagram? The idea of a fictitious link does not seem a good one. In a previous paper we rejected optional communication links and supported the idea of links that are not instances of associations [1]. However, we are not satisfied with this solution, since a link, like an object, is a "concrete thing" (an instance); thus, a link, like an object, requires a "type" that specifies its features; the type of an object is a class, and the type of a link should be an association that specifies, among other features, the classes of the linked objects, the navigability and changeability of the links, etc. If a link had no type, we would not be able to describe which its properties are or how it is supposed to behave[3]. Therefore, we will try to find a conceptually better solution that avoids "baseless links".

3. Association and link stereotypes in the UML Standard

Some authors have tried to distinguish between two disjoint subtypes of associations to solve this problem, "static associations" and "dynamic associations" [3], but we consider that this distinction is not adequate, since in object orientation every association has both static and dynamic features.

If static/dynamic is not an adequate classification of associations, how can we distinguish "normal" associations from other kinds of associations that seem relevant in modeling? UML has five predefined stereotypes for links ends (in the metamodel, `LinkEnd` metaclass) which are supposed to solve the dynamic associations issue, that is, how an instance can communicate with another instance without any existing (static) association between the respective classes. The five stereotypes specify different ways in which an instance is "visible"[4] [UML, p. 2-103]:

- «`association`»: the instance is visible via association.
- «`global`»: the instance is visible because it is in a global scope relative to the link.
- «`local`»: the instance is visible because it is in a local scope relative to the link.
- «`parameter`»: the instance is visible because it is a parameter relative to the link.
- «`self`»: the instance is visible because it is the dispatcher of a request.

For association ends (in the metamodel, `AssociationEnd` metaclass), we have the same five stereotypes, although their definitions are slightly different [UML, p. 2-24]. It is worth to copy them here and compare with the preceding ones, which are rather obscure:

- «`association`»: specifies a real association; default and redundant option, although it can be used for emphasis.
- «`global`»: the target is a global value known to all elements, rather than an actual association.
- «`local`»: the relationship represents a local variable inside the procedure, rather than an actual association.
- «`parameter`»: the relationship represents a procedure parameter, rather than an actual association.
- «`self`»: the relationship represents a reference to the object that owns the operation or action, rather than an actual association.

The intention of the Standard in defining these five stereotypes is not very clear. On the one hand, it seems that we should have a *coherence rule*, in the sense that a link end having a certain stereotype implies the same stereotype for its corresponding association end; but the Standard does not impose this restriction. On the other hand, the four stereotypes «`global`», «`local`», «`parameter`», and «`self`» are apparently intended to give a kind of access that is not properly derived from an association, but from other circumstances, supporting the statement that there are communication links which are not instances of associations; that is, a stereotyped link end would correspond to no association end, not even to one having the same stereotype. But this would contradict the suggested coherence rule, and make the stereotypes unnecessary for association ends (that is, they would be required for link ends only)[5].

Maybe this paradox is due to a careless writing of the Standard, rather than to a true inconsistency. We may suppose that the intention was to state that every link is an instance of an association, but there are "special" links that are not instances of normal associations, but special *implicit* associations, which do exist without need of being declared in the model, although the modeler can declare them in favor of clearness.

[3] There are some object-oriented *untyped* programming languages, such as Eiffel or Smalltalk, and even though it is completely legitimate the use of UML to model systems implemented in these languages, UML itself is strongly typed, likewise languages such as Java.

[4] Note how the Standard is imprecise in using the concept of visibility in these definitions. Instead of "visible", it should say "accessible".

[5] We have already mentioned in Section 2 another contradiction in the Standard, when it states that the communication link is not necessary in certain special situations (the same ones in which these stereotypes are defined) [UML, p. 2-114].

4. A proposed solution

In this Section we briefly propose a solution that reaffirms the principle that every link is an instance of an association, and uses association and link stereotypes consistently.

Associations

- Every association must be declared in a well-formed model, in order to specify its features and to avoid interactions that might be inconsistent with the rest of the model.
- It is not necessary that every association appears in a class diagram. It is enough that the association is represented in the underlying model.
- The different kinds of associations are distinguished by the stereotypes applied.

Links

- In a well-formed model, every link is an instance of an association.
- During the initial phases of a model's development, it is legal to represent links without specifying its association.
- A link bears the same stereotype as its association.
- Every stimulus or message instance requires a communication link.

This proposal implies a correction in the multiplicity of the `Stimulus.communicationLink` role [UML, p. 2-98] (change from 0..1 to 1..1).

5. Conclusions

CASE tools could be designed to help modelers in avoiding the baseless links problem. Probably, it is not convenient, in general, to require in a model that every link that appears in an object or collaboration diagram must correspond to an association in a class diagram. In our approach, every link is an instance of an association, but you don't need to show every association in a diagram: it is enough that these associations are represented in the underlying model, even though they do not appear in any diagram. It is convenient that the tool allows (even requires) the specification of the corresponding associations, likewise a class is specified for every object[6]: in this way you avoid that the association properties remain unspecified or that the modeler specifies an interaction that is inconsistent with the rest of the model.

References

1. Gonzalo Génova, Juan Lloréns, Vicente Palacios. "Sending Messages in UML", *Journal of Object Technology*, vol.2, no.1, Jan-Feb 2003, pp. 99-115.
2. Object Management Group. *Unified Modeling Language Specification*, Version 1.4, September 2001 (http://www.omg.org/).
3. Perdita Stevens. "On the Interpretation of Binary Associations in the Unified Modelling Language", *Journal on Software and Systems Modeling*, 1(1):68-79 (2002).
4. The Precise UML Group (http://www.cs.york.ac.uk/puml/).

[6] In certain development phases of a software project, especially in the initial ones, it is convenient not to specify an object's class, and in the same way the specification of a link's association should not be mandatory. CASE tools can enable or disable this feature as it is convenient for the modeler.