Modifying Object-Role Modeling into Situated-Object-Behavior Modeling with Metapattern

Pieter Wisse

Some modifications of Object-Role Modeling (ORM) make it equivalent with Metapattern. Metapattern's key assumption is that behavior is exhibited by situated objects, only.[1] So, starting from X as a situation and Y as an object, as in figure 1.a, a situated object Z may result (figure 1.b). In figure 1.c the descriptive labels have been removed. Instead, the relationship is now shown as directed. The arrow does not indicate more than that the parts, indeed, roles of situation and object are distinguished in constituting a situated object.

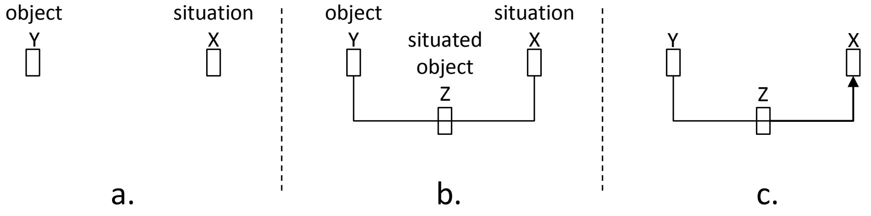

figure 1: establishing a situated object for behavior attribution.

With Metapattern, anything may be taken as either situation or object for deriving a situated object with correspondingly particular behavior. And, from an opposite perspective, each situated object must have originated from both 'its' predifferential object and the situation relevant for behavioral differentiation.
In order to avoid an infinite regression, a boundary condition holds. The so-called horizon, drawn as a solid line at the top of a conceptual model, is a situated object which is constituted by itself as both situation and object.[2]

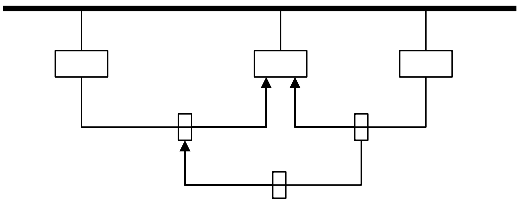

figure 2: how Metapattern models typically look.

[1] The method for conceptual modeling, later to be called Metapattern, was first described by me in Dutch in 1991. An English translation appeared as *Multicontextual paradigm for object orientation: a development of information modeling toward fifth behavioral form* (in: *Informatiekundige ontwerpleer*, Ten Hagen Stam, 1999). See also my book *Metapattern: context and time in information models* (Addison-Wesley, 2001). Metapattern's notation was partly changed in 2002 making it easier to express and interpret cascading situatedness; for an overview see leaflet, *Metapattern, development of notation* (Information Dynamics, 2012).

[2] This is only one of three ways of putting it. Of course, other expressions start from situation and object, respectively.

And then there may be situated objects for which the horizon acts as situation and/or object. (Relationships in which the horizon is involved need not display direction at all.) An abstracted, very small model, yet already typical of Metapattern, is shown in figure 2.

ORM's concept of a fact is that of objects related according to roles particular to that fact/relationship.[3] Here, we only need to consider what ORM views as an elementary fact, that is, a fact involving a binary relationship; see figure 3.

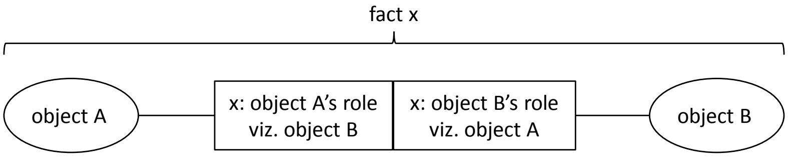
fact x
object A
x: object A's role viz. object B
x: object B's role viz. object A
object B

figure 3: ORM's elementary fact as a binary relationship.

What ORM calls an association appears to be limited to what figure 3 shows as the rectangle with compartments for the roles contributing to the overall relationship.

Applying Metapattern basically requires the following constraint. One of the objects constituting ORM's relationship changes into situation. In figure 4, therefore, figure 1.b is represented the ORM way.

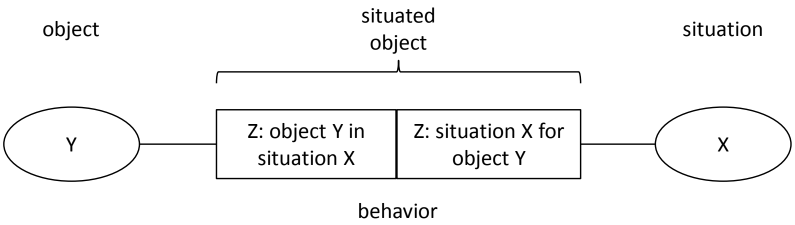

figure 4: starting the Metapattern turn with ORM.

ORM does provide for the possibility to objectify an association. As such, it becomes yet another object available for describing facts.
When Metapattern constitutes a situated object, such objectifying is inherent. In terms of modified ORM, therefore, an association is available in principle to act as either object or situation for additional relationships with associations.

How ORM represents an objectified association is by including the original compartmentalized rectangle in a larger rectangle. However, ORM's different symbols for, say, original object and object-derived-from-association, respectively, seem detrimental to abstraction and its expressive power. We'll progress from figure 4 to add modifications.

In figure 5, the association is objectified still using ORM's surrounding rectangle. What has been changed is removing situation and object from the level of association a.k.a. situated

[3] ORM is well documented. For example, see *Conceptual Schema & Relational Database Design* (WytLytPub, 1999) by T.A. Halpin.

object. This ordering is merely practical as it helps keeping more elaborate models easier to both design and review.

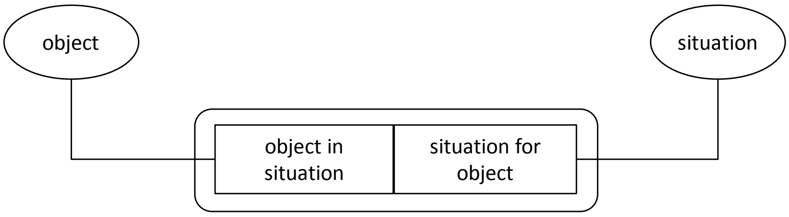

figure 5: making a principle out of objectifying association.

Next, figure 6 proposes the same symbol for everything that may be taken as situation and/or object. This includes association! For example, either association is drawn as an ellipsis, too, as in figure 6.a, or the shape for object and situation is changed into a rectangle (see figure 6.b).[4]

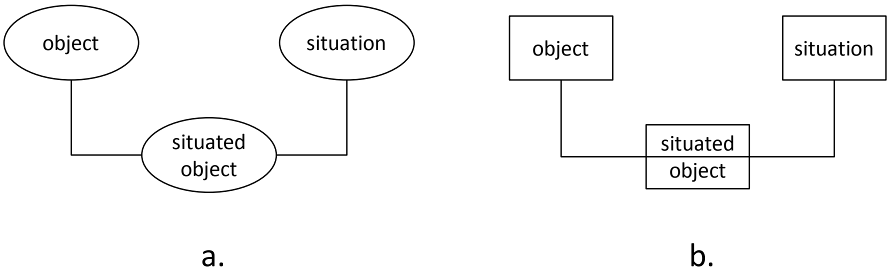

figure 6: towards unified symbolism.

My favorite is the rectangle, for reasons of both aesthetics and economy. The next step does not need to be drawn out, not again. See figure 1.b.
Please note that in figure 6.b, too, the line connecting the original situation and object traverses the rectangle that stands for the resulting situated object. It is Metapattern’s way of expressing that a relationship is object and, the other way around, an object is relationship.

Of course, so far the main question has not been dealt with. Has perhaps something gone missing from ORM on its way to Metapattern?
On the contrary, orienting an object’s behavior through situation rather than at one or more other objects directly introduces an explicit invariant regarding all the objects involved. It is what makes Metapattern flexible, indifferent to modeling scope (also read: universe of discourse) and therefore more realistic.

With ORM, when the role of one of the participating objects changes, the whole association is invariably committed. Metapattern limits what needs to be adjusted to the particular situated object, only. What Metapattern optimally helps to avoid are changes in information and its structure for which no actual events have occurred as their cause. The paradigm of situated

[4] Halpin already did consider precisely these notational alternatives, but decided for different symbols as he explains in *Conceptual Schema & Relational Database Design* (p. 97).

objects and their unambiguous behavior keeps information in as tight as possible correspondence with … facts.

Figure 7 summarizes how ORM develops into Metapattern by superimposing a situation on a relationship as originally modeled with ORM. What is shown here for ORMs binary relationship, of course holds in general, i.e. for an n-ary relationship.

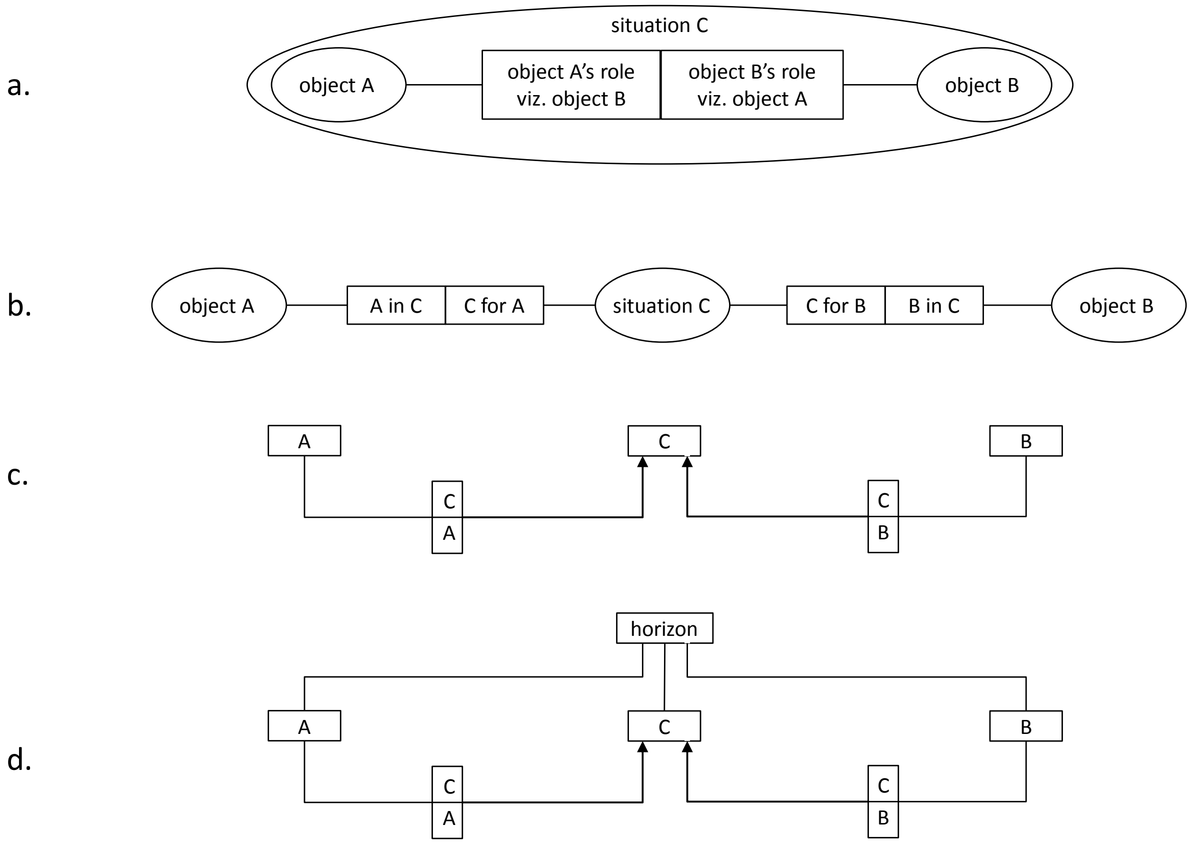

figure 7: situate and objectify, both outward and inward.

Take a relationship modeled with ORM, as in figure 3. Looking outward, superimpose a situation;[5] see figure 7.a. This situation may also become, so to speak, objectified. In figure 7.b, indeed, the direct association between objects is replaced, for each object involved, by a relationship with the situation. What follows in figure 7.c is that the associations are objectified, too, while unifying symbols. Figure 7.d adds the horizon ensuring that the conceptual model only displays situated objects. Please note how structurally similar figure 7.d is to figure 2; the only difference is that at the bottom of figure 2, two nodes are connected adding a node as representing yet another situated object.

[5] What may usually be considered a constraint from the perspective of more traditional languages/methods for conceptual modeling, should motivate establishing a situation from Metapattern's modeling perspective. That is, Metapattern removes to a large extent the need for constraints being stipulated, say, after the fact. With Metapattern, superimposing a situation as a set of behavioral preconditions is already taking care of being specific in a both necessary and sufficient manner.

This, say, proof that with proper modifications models drawn up according to ORM can become Metapattern models, implies that the same goes for a.o. Entity-Relationship Modeling and Object Orientation.[6] For whatever can first be expressed by ORM, may subsequently be expressed by Metapattern. To summarize, it works through superimposition of a situation and changing the thus situated n-ary ORM relationship into n binary ORM relationships with the constraint that the situation in question is ‘object’ to each of those binary associations.

At the bottom of the difference between Metapattern and ORM is a shift in understanding relativity. In a way, ORM still clings to so-called logical atomism, where an atom (also read: object) is believed to exist first of all independently. Metapattern assumes radical interdependency.[7]

About the author: Pieter E. Wisse (www.wisse.cc) is founder and president of Information Dynamics, an independent company operating from the Netherlands and involved in research & development of complex, civil information management. Pieter holds an engineering degree (mathematics and information management) from Delft University of Technology and a PhD (information management) from the University of Amsterdam.

[6] Direct transformation between an entity-relationship (ER) model or an object-oriented model (OO) on the one side, and a Metapattern model on the other, of course only works so far as attributes remaining inside an entity/object are not included. It is also explained for ER in appendix C, section C.9, of *Semantiek op stelselschaal: issues en oplossingsrichtingen* (Office of the Standardisation Forum, The Netherlands, 2009). That report is written in Dutch and addresses semantic interoperability through Metapattern, with appendix C providing an evaluation of Metapattern by an independent party. As the evaluator had no previous knowledge of and experience with Metapattern, I suggested from my earlier work (e.g. see *Metapattern: context and time in information models*, pp. 117-118) that the focus with the transformation should lie on the so-called weak entity. Please note the correspondence between weak entity (ER) and association (ORM), and that weak entity with ER is precisely what Metapattern would call strong.

[7] For an introduction, see my essay *Ontology for interdependency: steps to an ecology of information management* (in: **PrimaVera**, working paper 2007-05, Amsterdam University, 2007).