Position paper for the REA workshop Spring 2004

Typification in REA

Signe Ellegård Borch
IT University of Copenhagen

The notion of type images and policies, and the distinction between knowledge level and operational level are important aspects of the REA ontology as presented in [GeertsMcCarthy2000]. This position paper will raise a number of questions on how to implement these aspects of the ontology.

The extension of the REA ontology with type images, and relations between these, has a foundation in philosophy – the ontology article refers to John Sowas categories of the abstract and the physical [Sowa99]. As such the extension fits the ontology approach well, as it makes it possible to categorise all REA concepts according to Sowas conceptual terminology [GeertsMcCarthy2002].
The partition of the REA model into an operational level and a knowledge level is inspired by Martin Fowler, who is using this pattern extensively in his book "Analysis Patterns"[Fowler97]. The idea of the knowledge level is that it mirrors the structure of the operational level, and functions as a sort of configuration mechanism for the operational level.

The operational/knowledge level pattern resembles the item descriptor pattern introduced by Peter Coad in 1992 [Coad92]. It is also known as Power Type [MartinOdell95], and Type Object Pattern[JohnsonWoolf97]. They all substitute the class/instance relationship with a type instance/instance relationship, so that the type of the instance no longer is defined by its class, but by what type instance it refers to.
The strength of these patterns is that they make dynamic grouping of instances and dynamic type change possible. One big difference between these patterns and Fowlers knowledge level/operational level is that the knowledge level is not only used as a type system - it is also used to define a so called policy infrastructure or abstract information structure. In [GeertsMcCarthy2002] the policy level is defined as follows:"The policy infrastructure on the other hand conceptualizes what "could be" or "should be" within the context of a defined portfolio of firm resources and capabilities".
The policies are expressed as relations – in the knowledge level they express the legal configurations of the operational level. It is possible for entities at the knowledge level to have more relations than the entities on the operational level, because the latter expresses the actual situation, and not the space of possible solutions.

The knowledge level thus functions as a meta-architecture above the operational level. It has a lot in common with the Active Object Model [LooteYoder98], which is a reflective architecture with object instances (not classes) representing the types of the objects on the operational level – if the object model (of the knowledge level) changes, the system will change its behaviour. It is thereby possible to adapt the system without recompiling it. In [NakamuraJohnson98] an implementation of a REA framework based on the Active Object Model is presented, but this was before the ontology article, and it defers somewhat from the ontology, e.g. it does not say anything about policies..

Why extend the original REA model with type images? One primary reason is that this makes it possible to state policies concerning the runtime system of REA. The level thinking is not an inherent part of the basic REA model – it is a convenient pattern for implementing REA, and making the systems based on REA flexible, but it doesn't affect the original REA model in any way. Thus, it belongs to the "implementational issues" of the model. It could be a subject of discussion whether the knowledge level/operational level aspects should be a part of the ontology at all.

One problem that must be taken into consideration when implementing REA is the question of how generic the solution should be when it comes to types. Even though the basic REA model only consists of the entities Resource, Event, and Agent, most of the concrete examples of the model has named instances of these entities (e.g. SalesPerson (an Agent), CashReciept (an Event), Rental (an Event), Car (a Resource)). It is not clear if these names are to be understood as types, or if they are just names given to the basic entities for the sake of the example. This problem is reappearing in all REA papers.
There seem to be some inconsistencies in the REA ontology article as to how the type images are presented and explained. The example in figure 5 [Geerts McCarthy 2000] shows the type images in a very generic way: the operational level shows the basic entities of REA – Resources, Events, Agents, and Commitments, and the knowledge level shows their corresponding type images ResourceType, EventType, AgentType, and CommitmentType and the relations between these. The basic entities are not typed themselves. Similar, the paragraph that explains the concept of type images gives examples of AgentType, EventType, ResourceType, and CommitmentType. However, the example on figure 6, shows an example with two (typed) Agents: Customer and SalesPerson. The SalesPerson is mapped to the type image SalesPersonType. In this example, SalesPersonType is about the skills of the SalesPerson (the level of experience). The difference in the level of generality of the two examples regarding the types leads to one important question:

Why distinguish between the types at the operational level, and the type images on knowledge level? It would make a lot of sense to handle all type aspects in a uniform way. In other words, does it make sense to distinguish between roles and types in the REA ontology? (where roles denotes the dynamic aspects, and types denotes the static aspects). Would it be possible to move all type information to the knowledge level, leaving only the basic REA entities at the operational level? What would be the trade-offs of doing so?

To make assumptions of how dynamic the REA type system should be, certain questions might be useful:

- can a specific instance of an Agent be both a customer and a salesperson at the same time? It somehow makes sense that the same Agent can be both inside and outside in a transformation scenario. In this case, typification on the operational level might be an obstacle.
- is the typification on the knowledge level something that is nested in a hierarchy(like configuration, where one choice opens up for more possible choices)? Or has it only got two levels?
- is there a one-to-many relationship between an object instance on the operational level and a type image on the knowledge level, or is it possible to categorise the same (operational) instance according to different type images (many-to-many)? (the examples in the ontology article only show one-to-many).

References

[Coad92] Coad, P. :*"Object-oriented Patterns," Communications of the ACM* . 35(9):152-159, September 1992.

[FooteYoder98] Foote, B. and Yoder, J.: *Metadata and active object models*. In OOPSLA '98 *MetaData and Active Object-Model* Workshop, Vancouver, Canada, October 1998.

[Fowler97] Fowler, M.: *Analysis Patterns: Reusable Object Models* . Addison-Wesley, Reading, MA, 1997.

[GeertsMcCarthy2000] Geerts, G. L., McCarthy, W. E.: *The Ontological Foundation of REA Enterprise Information Systems*, 2000

[GeertsMcCarthy2002] Geerts, G. L., McCarthy, W. E.: *An Ontological Analysis of the Economic Primitives of the Extended REA Enterprise Information Architecture*, International Journal of Accounting Information Systems, No.3, 2002.

[JohnsonWoolf97] Johnson, R. , Woolf, B.: "*Type Object*," Pattern Languages of Program Design 3, Robert Martin, Dirk Riehle, and Frank Buschmann, eds., Addison-Wesley, Reading, MA., 1997.

[MartinOdell95] Martin, J., Odell, J.: *Object Oriented Methods: A Foundation* . Prentice Hall, Englewood Cliffs, NJ, 1995.

[NakamuraJohnson98] Nakamura, H., Johnson, R.: *Adaptive Framework for the REA Accounting Model*, OOPSLA’98, Workshop on Business Object Design and Implementation IV, 1998.

[Sowa99] Sowa, J.: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks/Cole Publishing, 1999.