Levels for Conceptual Modeling

Claudio Masolo

Laboratory for Applied Ontology, ISTC-CNR, Trento, Italy
masolo@loa-cnr.it

Abstract. Usually object types are organized in taxonomies by means of a specialization relation (also called subtyping or isa) ‘implemented’ by means of inheritance. This paper proposes a (non-incompatible) alternative to taxonomies that relies on three primitives: grounding, a specific kind of factual existential dependence, extensional atemporal parthood, and existence at a time. On the basis of these relations, specific, generic, and compositional grounding relations between object types are introduced. By clearly separating the objects from the substrata on which they are grounded, these grounding relations allow to stratify object types in levels and to manage inheritance in a flexible way. In particular, this approach helps to avoid isa overloading and to overcome some classical difficulties related to inheritance, e.g. attribute overriding, attribute hiding, or dynamic and multiple classification and specialization, that are relevant aspects especially in modeling roles.

Keywords: Grounding, Dependence, Levels, Taxonomies, Inheritance.

Classification schemes – taxonomies based on subtyping (isa) among object types – and inheritance are central notions in conceptual modeling (CM) and in object-oriented modeling. By assuming, for instance, that Statue is a subtype of Amount Of Matter, Statue inherits all the attributes and associations of Amount Of Matter. However, new attributes can be introduced. For instance, Statue, but not Amount Of Matter, could have the attribute Style. Similarly roles like Student, Customer, or President could be modeled as subtypes of Person. Student, but not Person, has a Matriculation. Customer, but not Person, has a Code, etc. This powerful mechanism of inheritance faces some well known problems. Statue and Amount Of Matter could have different values for the ‘same’ attribute, e.g. Price: a statue could be more expensive than a brute piece of bronze. Customer, differently from Person, could not have Weight or Place Of Birth. Attribute overriding and hiding try to manage these problems. Furthermore, roles can be played by objects with ‘incompatible’ attributes. For instance, both companies and persons can be customers, but Customer is neither

---

1 Amounts of matter are concrete entities, pieces of matter, specific sums of molecules, e.g. the piece of gold that now constitutes my ring, not ‘the gold’ or ‘gold’.
2 In this paper I will focus only on data modeling and not on behavior modeling.
3 The term role indicates here a specific kind of object types (properties). Roles intended as ‘parts in relationships’ are close to relational roles (see [12]).
a subtype nor a supertype of both Company and Person. “[W]e have the paradoxical situation that, from the extensional point of view, roles are supertypes statically, while dynamically they are subtypes” ([17], p.90). While keeping the same domain, this problem can be managed by adding new objects types, e.g. Private Customer (subtype of both Person and Customer) and Corporate Customer (subtype of both Company and Customer) [7], or by introducing dynamic and multiple classification and specialization (see [17] for a review). Alternatively, more permissive or multiplicative approaches extend the domain with new entities. Steimann [17] separates natural types (e.g. Person) from role types (e.g. Customer). Roles are adjunct instances linked by a played-by relation to their players (the persons or companies in the case of customers). The object and its roles form an aggregate and “the dynamic picking up of a role corresponds to the creation of a new instance of the corresponding role type and its integration in a compound, and dropping a role means releasing the role instance from the unit and destroy it” ([17], p.91). In object-oriented database management systems, by distinguishing specialization, an abstract concept, from inheritance, a mechanism that implements specialization, [1] systematically multiplies the instances in the presence of a subtype relation. If \( P \) is a subtype of \( Q \), then the creation of an object \( p \) of type \( P \) produces the creation of an object \( q \) of type \( Q \) plus a link between them that allows \( p \) to inherit attributes from \( q \). An object then is implemented “by multiple instances which represent its many faceted nature. Those instances are linked together through aggregation links in a specialization relation” ([1], p.561). The attributes are locally defined and stored but additional ones can be inherited via the links between the instances. From a more foundational perspective, multiplicative approaches have been investigated to solve the counting problem [9]. For instance, to count the Alitalia passengers (during 2010), one cannot just count the persons that flew Alitalia (during 2010). By adding qua-entities [12], (sum of) relational tropes [7], or role-holders [13] – entities that inhere in (a sort of existential specific dependence), but are different from, the players (see Section [1] for more details) – the counting problem is solved. In philosophy, multiplicativism is often considered also in the case of statues, organisms, tables, etc. (see [14] for a review and [3] for a recent defense). Interestingly, qua-entities have been originally introduced in this contest [6]. As in the case of roles, statues and amounts of matter have different properties (in particular causal properties) and different persistence conditions. The amount of matter that constitutes a specific statue can change through time. Or, an amount of matter can constitute some statue only during a part of its life, when it is statue-shaped. Therefore, some authors assume that statues are constituted by (a sort of existential dependence), but different from, amounts of matter.

Taxonomies are undeniably an important conceptual tool to organize object types according to the set-theoretical inclusion between their extensions. But it is not the only one. This paper proposes a complementary structuring mechanism founded on a specific kind of existential dependence called grounding. This mechanism allows to account for both roles and material objects with a flexible management of inheritance that helps to avoid isa overloading and misuse.