Toward an ontology-driven unifying metamodel for UML Class Diagrams, EER, and ORM2

C. Maria Keet[1] and Pablo Rubén Fillottrani[2,3]

[1]School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal and UKZN/CSIR-Meraka Centre for Artificial Intelligence Research, South Africa, keet@ukzn.ac.za
[2] Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, prf@cs.uns.edu.ar
[3]Comisión de Investigaciones Científicas, Provincia de Buenos Aires, Argentina

Abstract. Software compatibility and application integration can be achieved using their respective conceptual data models. However, each model may be represented in a different language. While such languages seem similar yet known to be distinct, no unifying framework exists that respects all of their language features. Aiming toward filling this gap, we designed a common, ontology-driven, metamodel of the static, structural, components of ER, EER, UML v2.4.1, ORM, and ORM2, such that each is a fragment of the encompassing consistent metamodel. This paper presents and overview and notable insights obtained on the real common core entities and constraints, roles and relationships, and attributes and value types that we refine with the notion of dimensional attribute.
Keywords: Metamodel, UML, EER, ORM

1 Introduction

Complex software system design and information integration from heterogeneous sources is required due to, among others, upscaling of scientific collaboration in the life sciences [33], e-government initiatives [29], company mergers [4], and the emergence of the Semantic Web. Therefore, establishing connections between multiple conceptual models has become an important task, as the system's conceptual data models are available in, mainly, UML, EER, or ORM. However, traditional information systems development and management only exhibit this capability at the physical schema layer [7] or for conceptual models represented in the same language [2,12]. Several works has been done for conceptual data models lately, but subtle representational and expressive differences in the languages makes this task very difficult, and current tools offer only very limited functionality in linking or importing models represented in one language into one represented in another language; e.g., mandatory and disjointness is catered for, but not weak entity types, identification, or attributes [7,8,9].

The differences between the main conceptual data modelling (CDM) languages—UML Class Diagrams, ER, EER, ORM, and ORM2—may seem merely terminological, but it is known that from a metamodelling viewpoint,

this is not the case [18], and at times not even within the same family of languages [25]; conversely, what may seem different may actually not be, or at least have a common 'parent' in meaning. The latter concerns differences in ontological foundations, but the state of the art in this area has not gone beyond a single CDM language and only for UML and ORM (e.g., [15,25]). Thus, it is unclear to what extent the languages differ or agree on their underlying principles for modelling information. This gap, in turn, is a limiting factor of mapping and transformation algorithms for CASE tools to let one work in parallel on conceptual data models represented in different languages that otherwise could be highly useful in information integration and complex system development. In addition, more detailed insight in the overlap and differences in underlying modelling principles will contribute to the understanding of the extent to which the language features affects modelling information as accurately as possible or needed, and to tools and methodologies for model development and maintenance.

To solve these issues, a comprehensive formalisation of the languages is needed to manage their interaction, but to arrive there, it first should be clear what entities and constraints exist in each language and how the differences can be reconciled without changing the languages. That is, not a comparison of metamodels, but a single integrated metamodel inclusive of all language features, so that one can unify the CDM languages and design straight-forward transformation algorithms at the conceptual layer in software and database development. We designed such a unifying metamodel for the static, structural components of UML 2.4.1 class diagrams, EER, and ORM2/FBM and their constraints, which, to the best of our knowledge, is the first of its kind. This metamodel is ontology-driven in the sense that our arguments are supported by insights from Ontology and ontologies rather than the argument of convenience to fit with an *a priori* chosen logic language. The unification brings afore the differences and commonalities, such as that they all agree only on Relationship (association), Role (/association end/relationship component), and Object type (/class/entity type), but also 'incomplete' coverage of certain features in one language that are present in full in another one, such as attributes. While for the static, structural, entities ontology helped the harmonization, this is much less the case for the constraints: jointly, there are 38 constraints, but there is a remarkable small overlap among the languages (mandatory, uniqueness/functional and cardinality constraints in general, disjointness and completeness, and subset constraints). In this paper, we provide a summary of the metamodel and, due to space limitations, discuss the two most salient and ontologically interesting aspects of it, being the roles and relationships, and attributes. We discuss related works in Section 2, introduce and describe some interesting design decisions in the metamodel in Section 3, and we conclude in Section 4.

2 Related works

There are different strands of investigation in different subfields that consider multiple CDM languages; the physical schema layer and integrating conceptual

models represented in the same language have their own issues and solutions, which is beyond the scope. Comparing the languages through their metamodels (in ORM) highlighting their differences [18] is a useful step before unifying them. The Unifying Foundational Ontology (UFO) focuses on the philosophical aspects of conceptual modelling and has been applied to extend the UML 2.0 metamodel with more specific entities, such as a Sortal Class [15,17]. Their ontological analysis of, among others, the nature of UML's class and association did not translate into a revised metamodel specification, however.

Venable and Grundy designed [37] and implemented a partial unification in MViews [13] and Pounamu [38]. Their metamodel in the CoCoA graphical language covers a part of ER and a part of NIAM (a precursor to ORM), and omits, mainly, value types, nested entity types, and composite attributes, and NIAM is forced to have the attributes as in ER in the 'integrated' metamodel. Consequently, their "dynamic" ad hoc mappings are limited.

Bowers and Delcambre [7] present a framework for representing schema and data information coming from several data models, mainly relational, XML and RDF. Its main characteristic is a flat representation of schema and data, and the possibility of establishing different levels of conformance between them. However, its representational language ULD only includes ordinal, set and union class constructs, and cardinality constraints.

Boyd and McBrien [8] uses the Hypergraph Data Model to relate schemas represented in ER, relational, UML, and ORM, and includes transformation rules between them. Using graphs as intermediate representation has the advantage of providing a simple irreducible form for schemas that can be used to prove schema equivalence. The representational language includes inclusion, exclusion and union class constructs, and mandatory, unique and reflexive constraints. The combination of these types of constraints gives a rich language, but roles, aggregation, and weak entity types are missing.

Atzeni et al [2,3] describe an automatic approach that translates a schema from one model to another. They provide a small set of "metaconstructs" that can be used to characterize different models. These metaconstructs are entities (called "abstracts"), attributes (called "lexicals"), relationships, generalization, foreign keys, and complex attributes. Automatic translations between schemas are produced in the Datalog language, but translations from a rich representational language may require a sequence of such basic translations, if possible.

Thalheim [36] developed a framework for modelling layered databases, possibly integrating databases in different paradigms, such as OLAP systems and streaming databases. This type of database modelling is out of scope since we focus only on databases described with CDM languages.

Concerning unification by means of a single formalisation in a chosen logic, there are separate formalizations, which can be seen as prerequisites, and partial unifications, e.g., [1,5,9,19,22,23,27,32]. Their approach is, mainly, to choose a logic and show it fits sufficiently with one of the CDM languages, and perhaps due to this, different logics are used for different CDM languages, therewith still not providing the sought-after interoperability for either of the languages

or among each other. For instance, the Description Logic $\mathcal{ALUNI}$ is used for a partial unification [9] but *DL-Lite* and $\mathcal{DLR}_{\mathit{ifd}}$ for formalisation [1,5], which is incomplete regarding the features is supports, and ORM has features that render the language undecidable [24]. As such, they cannot simply be linked up and implemented. Once there is a comprehensive metamodel, they could be either formalised in one logic, or possibly the different logics (if implemented) could be orchestrated by means of the Distributed Ontology Language and system that is currently being standardised by ISO (`http://ontoiop.org`).

Our approach is different regarding two main aspects: scope and methodology. We aim to capture *all* the languages' constructs and *generalise* in an ontology-driven way so that the integrated metamodel subsumes the elements of EER, UML Class Diagrams v2.4.1, and ORM2 without changing the base languages. Such an integrated metamodel has as *fragments* the EER, UML Class Diagrams v2.4.1, and ORM2 metamodels, respectively, therewith leaving the base languages intact. None of the related works includes roles, aggregation, and relationship constraints, thus only limited subsets of UML or ORM are covered. Methodologically, our metamodel is ontological rather than formal, compared to all other known works that present first a formal common language for translations that leave aside important particular aspects of each language. We first develop a conceptual model of all possible entities and their relations in the selected languages, and will devise a formalization for their translations afterward. The main benefit is that it allows one to have a clear comprehension of the meaning(s) of an entity in each language whilst coping with the broader scope. This is an essential step towards achieving the full potential of information sharing.

3 Ontology-informed metamodelling

We focus on the *metamodel*, it being a conceptual model about the selected CDM languages such that this metamodel covers all their native features and is still consistent. Here, we do not question whether a feature of a language is a good feature or how one can make it better by using some ontological principles, but instead we aim at representing in a unified way what is present in the language. To achieve this, we use several notions from Ontology and ontologies, which serve to enhance understanding of the extant features, to reconcile or unify perceived differences, and to improve the quality of the metamodel. This does not make the metamodel an ontology, for its scope is still just the selected modelling languages.

The core entities are shown in Fig. 1 in UML Class Diagram notation and the overview of the constraints is included in Fig. 2, where a white fill of a class icon means that that entity is not present in a language, a single diagonal fill that it is present in one language, a double diagonal that it is present in two, and a dark fill that it is present in all three groups of languages (EER, UML v2.4.1, ORM2); naming conventions and terminological differences and similarities of the entities are listed in Table 1 at the end of the paper. The overview picture of the constraints contains only those that are explicitly available in the language as graphical or textual constraint in the diagram (note that OCL is a

separate OMG standard). Figures 3 and 5 are incomplete with respect to the full set of constraints that apply due to the limited expressiveness of UML Class Diagrams, but we preferred a more widely understood graphical notation for communication over a richer one, such as ORM, as it will be formalised in a suitable logic anyway. The reader may note there are a few redundancies in the metamodel; e.g., multivalued attributes can be represented by means of plain attributes. However, we aim to be complete with respect to the graphical features in the CDM languages, and for the metamodel not to judge whether it can be represented more elegantly (this can be addressed in a formalization). We describe several salient aspects of the metamodel and explain and motivate its contents in this section. More precisely, we discuss roles, relationship, and attributes (and omit from this discussion class/entity type, nested and weak entity type, subsumption, and aggregation); the metamodel for each constraint has been developed but is omitted from the discussion due to space limitations.

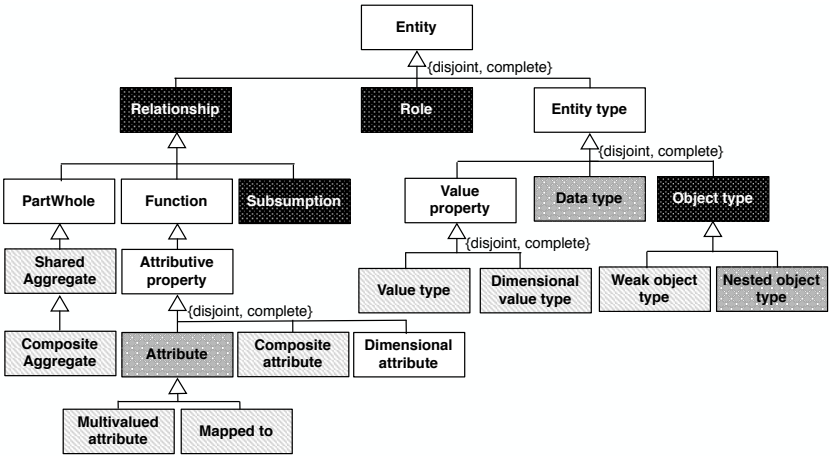

Fig. 1. Principal static entities of the metamodel; see text for details.

3.1 Roles and relationships

There are many points that can be discussed about roles and relationship, but we shall restrict ourselves to their nature and definition (not the possible types, as in [16]), and differences among them and with a object type; attributes will be discussed afterward.

A relationship, or relational property in ontology [34], is an entity that relates entities, hence, it requires at least two entities to participate in it, unlike an entity type that is a thing on its own. By this basic distinction, one can deduce that there are no unary relationships, in contrast to object types and unary predicates. The second difference between relationship and object type is due to *roles*, which are called "association ends" or "member ends" in UML [31],

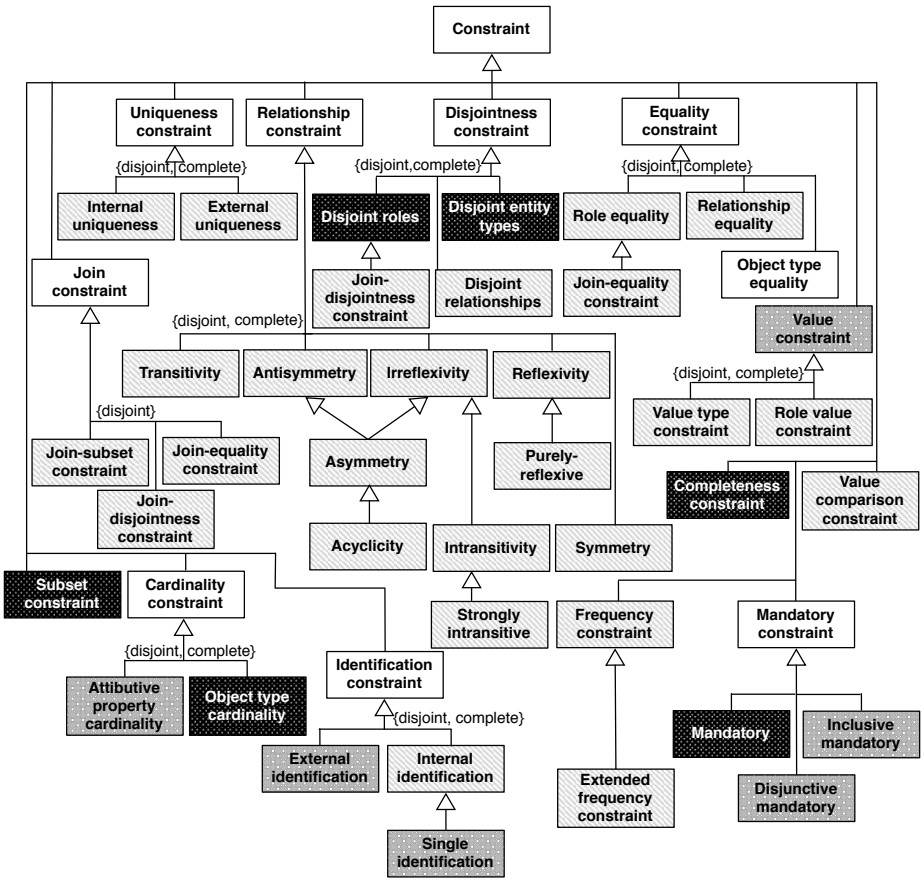

Fig. 2. Unified hierarchy of constraints in the metamodel; see text for details.

"roles" in ORM and fact-based modelling [19,20,11], components of a relationship in EER [10,35]. A role is something that an object plays in a relationship, and, thus, a relationship is composed of at least two roles, therewith contributing to the characterisation of relationships and committing to the so-called *positionalist* ontological commitment of relations and relationships (see [28] for a good overview, which has been applied to ORM in [25]). The consequences are that all three CDM languages have roles—hence, logically, they have to be first-class citizens in a formalisation—and, ontologically, this inclusion entails that from an information modelling viewpoint, they form a part of the so-called 'fundamental furniture of the universe' and thus that they are ontologically distinct from entity types and relationships. Therefore, they appear in the metamodel as separate entities (recall Fig. 1), and Relationship, Role, and Entity Type are disjoint. The interaction between them and predicates is depicted in Fig. 3. There are three points of note. First, the relationship between Role and Relationship is

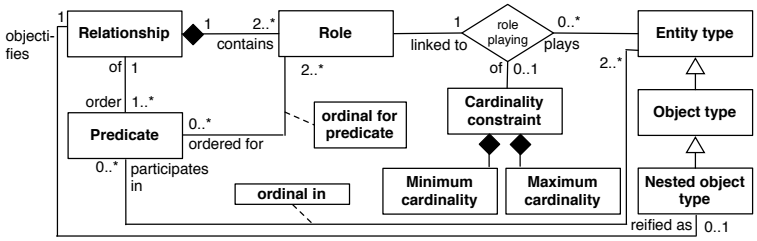

Fig. 3. Principal relationships between Relationship, Role, and Entity type; relations between roles and a predicate can only exist if there is a relation between those roles and the relationship that that predicate is an ordering of (i.e., it is a join-subset), and likewise that entities that participate in the predicate must play those roles that compose the relationship of which that predicate is an ordered version of it.

a composite aggregation. Ontologically, this might be considered an underspecification, as one may argue that a relationship is even *defined* by its role-parts. Second, Fig. 3 has a ternary role playing between Role, Entity type, and Cardinality constraint: each role must have exactly one entity type with no or one cardinality constraint (where the minimum and maximum cardinality are part of the Cardinality constraint), and each entity type may play zero or more roles with or without declared cardinality constraint.

Third, the inclusion of predicates. While each language has roles and names for relationships, only ORM adds predicates as another way to handle its fact types. In ORM, a relationship is composed of an unordered set of roles, whereas predicates have the participating objects ordered in a fixed sequence. Given the way relationships are used in CDM languages, we restrict Predicate to be at least binary in the metamodel, and likewise for Relationship. In logic, predicates can very well be unary, but unary predicates, ontologically, are of a different type, and, besides, order in an unary predicate does not make sense. The relationship between Relationship and Predicate is not clear. The draft ORM/FBM ISO standard depicts Predicate with a composite aggregation to Fact type (relationship) [11]. This is incorrect, because relationship and predicate each exemplify a different ontological commitment: predicate adheres to the so-called "standard view" and the latter to "positionalism" that requires the existence of roles [28], and each permutation of an ordering among participating objects without the use of roles is therewith not part of a single unordered composition of roles, and the predicates do not constitute the relationship (roles do). Yet, it is not subsumption either, because some thing without roles cannot be always a kind of something with roles. Therefore, it is included now as a plain association, where a predicate is one of the possible orderings of the entities that play the roles in that relationship. Likewise, one can argue that Role is not *ordered in* a, but *for* a, Predicate, because the roles are ordered and then 'removed' to obtain a predicate. Alternatively, one can look at it that roles have nothing to do with predicates, which is ontologically more precise, so that it is only Entity type that participates

in zero or more Predicates. While the idea is intuitively clear and sufficient for our current purpose, ontologically, the interaction deserves refinement.

Concerning subsumption, this is straightforward for entity types, but is less clear for relationships and roles. UML 2.4.1 distinguishes between subsetting and "specialization" of associations [31]. All CDM languages support subsetting of relationships, where the participating object types are sub-ends/sub-classes of those participating in the super-relationships (or indirectly through a relationship attributes). Specialization has to do with differences in intension of the association [31], but the UML standard does not describe how that is supposed to work. The only known option to change an association's intension, is to restrict the relational constraints [26]—e.g., each relationship that is asymmetric is also irreflexive—but little is known about its practicality, other than the few experiments in [26] for ontologies. To be comprehensive, both ways of relationship subsumption are captured in the metamodel with the more general Subsumption, and both UML and ORM include subsumption of roles, therefore, the participating entities for Subsumption are Entity (not shown).

3.2 Attributes, value types, and data types

From a formal perspective, it is clear what an attribute is, but this does not hold for CDM languages and ontologists have various theoretical frameworks to deal with them (albeit only partially), as discussed in [6,21]. We shall address the definition of an attribute, what Ontology and ontologies have to say about it, how it is incorporated in UML, EER, and ORM, the issue of the dimension, and how we represent it in the common metamodel and why.

Formally, an *attribute* (A) is a binary relationship between a *relationship* or *object type* ($R \cup E$) and a *data type* (D), i.e., $A \mapsto R \cup E \times D$. For instance, one can have an attribute hasColour, that relates an object type to a string; e.g, hasColour $\mapsto$ Flower $\times$ `String`. An attribute is no more, and no less.

'Attributes' in Ontology and ontologies Observe the disjointness between Object type, Value type, and Attributive property in the metamodel (Fig. 1), which reflects their meaning in the CDM languages and it also can be motivated by Ontology and ontologies. Ontology distinguishes between various kinds of properties [34] (provided one accepts the existence of properties, which the CDM languages do). Object type is in Ontology called, among others, a *sortal* property, which are, roughly, those things by means we distinguish, classify, and identify objects and they can exist on their own; e.g., Apple, Person. Attributions, or qualities, such as the Colour of an apple or its Shape, need a bearer to exist and are also formalised as unary predicates, not as attribute as in the definition above, and one cannot sort the objects by its attribution and know what those objects are (other than, e.g., 'red objects' or 'square objects'). Thus, attributions differ from sortals, and philosophers agree on this matter. Philosophers do discuss about the details of the attributions, however, such as whether they are universals or tropes or a combination thereof [6,14], which is relevant for linking the metamodel to

a foundational ontology, but we need not commit to either one of those for the metamodel. The distinction between object types and attributive properties is also reflected in the foundational ontologies; e.g., DOLCE has a distinction between Quality (the attribution) and Endurant/Perdurant (the object or relation) [6] and the GFO includes this ontological distinction (albeit different from DOLCE) and has refinements for dimensional and other attributes and between atomic and non-atomic attributes [21], where the latter is, in spirit, the same as EER's composite attribute.

Attributions in UML, EER, ORM Examples of the notations of attributions in the CDM languages are depicted in Fig. 4. UML class diagrams use the aforementioned definition and meaning of attribute and it is typically represented 'inside' a class- or association-icon as, e.g., "hasColour:String", although hasColour may be drawn also as an association with at the far end a class-icon for the data type [31]. ER and EER support an *incomplete* attribute specification that involves only the ER/EER entity type and the attribute name, noting that declaring datatypes is carried out at the physical design stage of the database development. This might give the impression that an attribute is an (unary, not binary) entity of itself, but, overall, the understanding from the formal foundation of ER and EER [10,35], is that the attribute is alike UML's attribute. ER and EER contain two additional types of attributes: composite and multivalued attributes. They have been included in the metamodel (see Fig. 1) because they are in the languages, but both formally and ontologically, there is no real addition, because both can be remodelled as basic attributes.

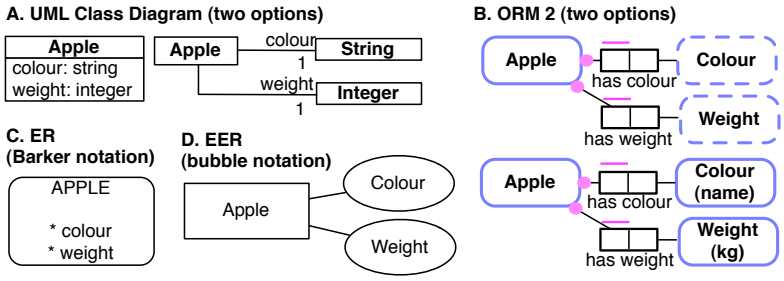

Fig. 4. Examples of attributes in UML Class Diagram notation, two ORM 2 options, and in two well-known ER/EER notations.

ORM is a so-called "attribute-free" language [20], yet, actually, they do have attributes in the strict sense of the meaning. It is true that ORM does not have attributes 'inside' the entity type, and what is modelled as a binary in UML, like the colour of the flower, is represented in ORM as a unary *value type* Colour that can be related to more than one entity type through a user-defined relationship (ORM fact type), but one has to specify the data type

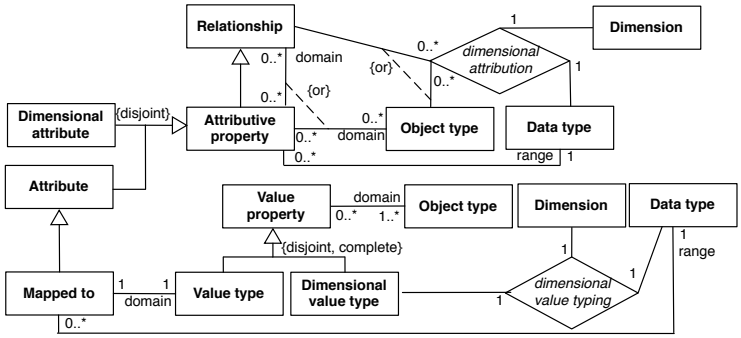

Fig. 5. Metamodel fragment for value properties and simple and attributes; Dimensional attribute is reified version of the ternary relation dimensional attribution, and likewise for Dimensional value type and dimensional value typing.

for the value type in the ORM diagram. An ORM value type's unique feature distinguishing it from an ORM entity type (in our metamodel, Object type), is that it has a behind-the-scenes "mapped to" relationship to the datatype [11]: mapped_to $\mapsto$ ValueType $\times$ `DataType`, which is generated by the software once a value type and its data type have been declared; for our example, we obtain a mapped_to $\mapsto$ Colour $\times$ `String` in addition to a binary relationship called, say, hasColour, asserted between Flower and Colour. Thus, ORM does have attributes in the strict sense of the meaning. The principal difference between UML's and ORM's attributes, is that ORM uses three entities with two binary relationships, whereas UML collapses it all into one binary relationship.

The CDM languages' attributes can be of relationships, of object types, or both. The latter may be contentious for UML and ER/EER, as one could argue that an attribute drawn in one Object type or Relationship is different from an attribute with the same name and data type that occurs in another Object type or Relationship. Practically in UML and ER/EER tools, one indeed has to add any subsequent occurrence of an attribute again, like hasColour:String not only for apples, but also for oranges, for cars and so on. However, there is no reason why one cannot keep a separate list of attributes for the whole model, and upon reuse, select an earlier-defined one; e.g., declaring hasHeight:Integer once, and use it for both Table and Chair. This is can be done already with OWL's data properties [30] in ontology editors, and one still has the option to add another height attribute, say, hasHeight:real when one has to make that distinction for another Object type. The alternative approach is to give each attribute a unique identifier and add a constraint that it must have exactly one domain declaration that is either an object type or a relationship, i.e, an {xor} instead of an {or} in Fig. 5. Practically, one still can do this in the tools, but we keep the less constrained, more flexible way of permitting attribute reuse in the metamodel just in case in the near future such more efficient and consistent attribute management is implemented in the UML and EER CASE tools.

The dimension ORM's CASE tools for modelling value types, such as in NORMA or VisioModeler, lets one add not only the data type, but also, if desired, the dimension of the measurement, such as cm, kg, or day, but there is no formal characterisation of it yet. Thus, besides the basic, composite, and multivalued attributes, there may be a *dimension* for the value, i.e., there is an implicit meaning in the values that has to do with measurements. For instance, when one needs to record the height of a plant, the measured value is not simply an integer or real, but actually denotes a value with respect to the measurement system (say, SI Units) and we measure in meters or centimeters. An attribute like hasHeight $\mapsto$ Flower $\times$ `Integer` does not contain any of that information, yet somehow one has to include that in the specification of the attribute or value type, even if just to facilitate data integration; e.g., as hasHeight $\mapsto$ Flower $\times$ `Integer` $\times$ cm, or as three relations: hasHeight $\mapsto$ Flower $\times$ Height, mapped_to $\mapsto$ Height$\times$`Integer`, and hasDimension $\mapsto$ Integer$\times$cm. Within the scope of unification, we are interested in adding the notion of dimension only, not how to represent a whole system of recording measurement data for a specific scenario, because these are subject domain notions for a specific application. Therefore we chose for the more precise ternary relation dimensional value typing in our metamodel, as shown in Fig. 5, with a mandatory data type and a dimension. It has its analogue for attributes (also shown in Fig. 5), although the solution is less obvious for UML, because the standard does not mention anything about dimensions. A ternary complicates the formal apparatus, but as unification is the aim, we prefer this option, therewith essentially allowing for an extension of UML's metamodel. While the figures for ORM and UML are different due to where the attribute resides in the language, the same principle is adhered to, i.e., using optional ternaries for dimensional attributes/value types.

4 Conclusions

We presented a summary of the unifying metamodel capturing ORM, EER, and static UML v2.4.1 conceptual data modelling languages with respect to their static, structural, entities and their relationships, and their constraints. In the strict sense of the languages' features meanings, the only intersection among all these CDM languages are role, relationship (including subsumption), and object type, with each therewith also adhering to the positionalist commitment of the meaning of relationship, and for the constraints disjointness, completeness, mandatory, object cardinality, and the subset constraint. Attributions are represented differently in each language, but, ontologically, they denote the same notions. Several implicit aspects, such as dimensional attribute and its reusability and relationship versus predicate, have been made explicit.

Due to space limitations, the details of the complete metamodel is not described; it has been developed already and revealed that, aside from identification, there are no crucial irreconcilable disagreements in the languages. We have commenced with the formalisation. Once completed, this metamodel will help in

the comprehension of differences between heterogenous conceptual models and in the development of tools that will aid information integration.

Acknowledgements This work is based upon research supported by the National Research Foundation of South Africa (Project UID: 80584) and the Argentinian Ministry of Science and Technology.

References

1. Artale, A., et al.: Reasoning over extended ER models. In: Proc. of ER'07. LNCS, vol. 4801, pp. 277–292. Springer (2007)
2. Atzeni, P., Cappellari, P., Torlone, R., Bernstein, P.A., Gianforme, G.: Model-independent schema translation. VLDB Journal 17(6), 1347–1370 (2008)
3. Atzeni, P., Gianforme, G., Cappellari, P.: Data model descriptions and translation signatures in a multi-model framework. AMAI Mathematics and Artificial Intelligence 63, 1–29 (2012)
4. Banal-Estanol, A.: Information-sharing implications of horizontal mergers. International Journal of Industrial Organization 25(1), 31–49 (2007)
5. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence 168(1-2), 70–118 (2005)
6. Borgo, S., Masolo, C.: Foundational choices in DOLCE. In: Handbook on Ontologies, pp. 361–381. Springer, 2 edn. (2009)
7. Bowers, S., Delcambre, L.M.L.: Using the uni-level description (ULD) to support data-model interoperability. Data & Knowledge Engineering 59(3), 511–533 (2006)
8. Boyd, M., McBrien, P.: Comparing and transforming between data models via an intermediate hypergraph data model. J. on Data Semantics IV, 69–109 (2005)
9. Calvanese, D., Lenzerini, M., Nardi, D.: Unifying class-based representation formalisms. Journal of Artificial Intelligence Research 11, 199–240 (1999)
10. Chen, P.P.: The entity-relationship model—toward a unified view of data. ACM Transactions on Database Systems 1(1), 9–36 (1976)
11. Committee Members: Information technology – metamodel framework for interoperability (MFI) – Part xx: Metamodel for Fact Based Information Model Registration (Draft release date: 2012-04-18 2012), iSO/IEC WD 19763-xx.02
12. Fillottrani, P.R., Franconi, E., Tessaris, S.: The ICOM 3.0 intelligent conceptual modelling tool and methodology. Semantic Web Journal 3(3), 293–306 (2012)
13. Grundy, J., Venable, J.: Towards an integrated environment for method engineering. In: Proceedings of the IFIP TC8, WG8.1/8.2 Method Engineering 1996 (ME'96). vol. 1, pp. 45–62 (1996)
14. Guizzardi, G., Masolo, C., Borgo, S.: In defense of a trope-based ontology for conceptual modeling: An example with the foundations of attributes, weak entities an datatypes. In: Proc. of ER'06. LNCS, vol. 4215, pp. 112–125. Springer (2006)
15. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Phd thesis, University of Twente, The Netherlands. Telematica Instituut Fundamental Research Series No. 15 (2005)
16. Guizzardi, G., Wagner, G.: What's in a relationship: An ontological analysis. In: Proc. of ER'08. LNCS, vol. 5231, pp. 83–97. Springer (2008)
17. Guizzardi, G., Wagner, G.: Using the unified foundational ontology (UFO) as a foundation for general conceptual modeling languages. In: Theory and Applications of Ontology: Computer Applications, pp. 175–196. Springer (2010)

18. Halpin, T.A.: Advanced Topics in Database Research, vol. 3, chap. Comparing Metamodels for ER, ORM and UML Data Models, pp. 23–44. Idea Publishing Group, Hershey PA, USA (2004)
19. Halpin, T.: A logical analysis of information systems: static aspects of the data-oriented perspective. Ph.D. thesis, University of Queensland, Australia (1989)
20. Halpin, T.: Information Modeling and Relational Databases. San Francisco: Morgan Kaufmann Publishers (2001)
21. Herre, H.: General Formal Ontology (GFO): A foundational ontology for conceptual modelling. In: Theory and Applications of Ontology: Computer Applications, chap. 14, pp. 297–345. Springer (2010)
22. Hofstede, A.H.M.t., Proper, H.A.: How to formalize it? formalization principles for information systems development methods. Information and Software Technology 40(10), 519–540 (1998)
23. Kaneiwa, K., Satoh, K.: Consistency checking algorithms for restricted UML class diagrams. In: Proc. of FoIKS'06. Springer (2006)
24. Keet, C.M.: Prospects for and issues with mapping the Object-Role Modeling language into $\mathcal{DLR}_{ifd}$. In: Proc. of DL'07. CEUR-WS, vol. 250, pp. 331–338 (2007)
25. Keet, C.M.: Positionalism of relations and its consequences for fact-oriented modelling. In: Proc. of ORM'09. LNCS, vol. 5872, pp. 735–744. Springer (2009)
26. Keet, C.M.: Detecting and revising flaws in OWL object property expressions. In: Proc. of EKAW'12. LNAI, vol. 7603, pp. 252–266. Springer (2012)
27. Keet, C.M.: Ontology-driven formal conceptual data modeling for biological data analysis. In: Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data, chap. 6, p. in press. Wiley (2013)
28. Leo, J.: Modeling relations. Journal of Philosophical Logic 37, 353–385 (2008)
29. Mendes Calo, K., Cenci, K.M., Fillottrani, P.R., Estevez, E.C.: Information sharing-benefits. Journal of Computer Science & Technology 12(2) (2012)
30. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C (27 Oct 2009), http://www.w3.org/TR/owl2-syntax/
31. Object Management Group: Superstructure specification. Standard 2.4.1, Object Management Group (2012), http://www.omg.org/spec/UML/2.4.1/
32. Queralt, A., Teniente, E.: Decidable reasoning in UML schemas with constraints. In: Proc. of CAiSE'08. LNCS, vol. 5074, pp. 281–295. Springer (2008)
33. Rosenthal, A., Mork, P., Li, M.H., Stanford, J., Koester, D., Reynolds, P.: Cloud computing: a new business paradigm for biomedical information sharing. Journal of Biomedical Informatics 43(2), 342–353 (2010)
34. Swoyer, C.: Properties. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Stanford, winter 2000 edn. (2000), http://plato.stanford.edu/archives/win2000/ entries/properties/
35. Thalheim, B.: Extended entity relationship model. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, vol. 1, pp. 1083–1091. Springer (2009)
36. Thalheim, B.: Model suites for multi-layered database modelling. In: Proceeding of the XXI Conference on Information Modelling and Knowledge Bases 2010. pp. 116–134. Frontiers in Artificial Intelligence and Applications, IOS Press (2010)
37. Venable, J., Grundy, J.: Integrating and supporting Entity Relationship and Object Role Models. In: Proc. of ER'95. LNCS, vol. 1021, pp. 318–328. Springer (1995)
38. Zhu, N., Grundy, J., Hosking, J.: Pounamu: a metatool for multi-view visual language environment construction. In: IEEE Conf. on Visual Languages and Human-Centric Computing 2004 (2004)

Table 1. Terminology comparison and conventions of the entities in UML Class Diagrams, EER, and ORM/FBM (for indicative purpose).

Metamodel	**UML v2.4.1**	**EER**	**ORM/FBM**
Relationship	association, can be 2-ary according to the MOF 2.4.1, but also >2-ary according to the Superstructure Spec 2.4.1	relationship, ≥2-ary	atomic/compound fact type, ≥1-ary
Predicate	absent	absent	predicate
Role	association end / member end	component of a relationship	role
Entity type	classifier	absent	object type
Object type	class	entity type	non-lexical object type / entity type
Attribute	attribute	attribute, but without including a data type in the diagram	absent (represented differently)
Dimensional attribute	absent (no recording of dimension)	absent	absent (represented differently)
Composite attribute	more general: a property can be a composite of another property (and an attribute is a property)	composite attribute	implicitly present by adding new roles
Multivalued attribute	absent (represented differently)	multivalued attribute	absent (represented differently)
Value type	absent	absent	lexical object type / value type, without dimension
Dimensional value type	absent	absent	lexical object type / value type, with dimension
Data type	Data type; LiteralSpecification	absent	data type
Object subtype	subclass	subtype	subtype
Sub-relationship	subsetting or subtyping of association	subtyping the relationship (not present in all EER variants)	subset constraint on fact type
Nested object type	association class	associative entity	objectified fact type
Composite aggregate	composite aggregation	absent	absent
Shared aggregate	shared aggregation	absent	absent