Data & Knowledge Engineering 98 (2015) 30–53

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2

C. Maria Keet [a,*], Pablo Rubén Fillottrani [b,c]

[a] Department of Computer Science, University of Cape Town, Cape Town 7701, South Africa
[b] Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina
[c] Comisión de Investigaciones Científicas, Provincia de Buenos Aires, Argentina

ARTICLE INFO

Available online 13 July 2015

Keywords:
Conceptual modeling
Ontologies
Metamodeling
EER
UML Class Diagrams
ORM

ASBTRACT

AsbtractSoftware interoperability and application integration can be realized through using their respective conceptual data models, which may be represented in different conceptual data modeling languages. Such modeling languages seem similar, yet are known to be distinct. Several translations between subsets of the languages' features exist, but there is no unifying framework that respects most language features of the static structural components and constraints. We aim to fill this gap. To this end, we designed a common and unified ontology-driven metamodel of the static, structural components, and constraints in such a way that it unifies ER, EER, UML Class Diagrams v2.4.1, and ORM and ORM2 such that each one is a proper fragment of the consistent metamodel. The paper also presents some notable insights into the relatively few common entities and constraints, an analysis on roles, relationships, and attributes, and other modeling motivations are discussed. We describe two practical use cases of the metamodel, being a quantitative assessment of the entities of 30 models in ER/EER, UML, and ORM/ORM2, and a qualitative evaluation of inter-model assertions.

1. Introduction

Recent upscaling scientific collaboration in the life sciences [1] and pharmacology[1], launching and monitoring of e-government initiatives [2,3], company mergers [4], integration frameworks [5], and a broader adoption of Semantic Web technologies require complex software system development and information integration from heterogeneous sources. While at the implementation and data level, markup languages and standards help interoperability (e.g., SBML [6], BEL [7], GML [8], RDF [9]), the design of such systems typically involves a data analysis stage with the design and linking or integration of conceptual models for the implementation-independent representation of the data requirements. The various extant conceptual data modeling (CDM) languages, such as UML [10], EER [11], ORM [12], MADS [13], and Telos [14], each have their strengths and typically multiple models represented in different languages are used for one system. Therefore, establishing connections between these conceptual models has become an important task.

A motivating use case to illustrate this is described in the following example, which describes one of the problems.

Example 1. Consider the interoperability scenario between an EER diagram and a UML Class Diagram where both are to be maintained as far as possible, which is depicted in Fig. 1. The EER diagram depicts a rudimentary model for a database of a generic

* Corresponding author. Tel.: +27 21 650 2667.
E-mail addresses: mkeet@cs.uct.ac.za (C.M. Keet), prf@cs.uns.edu.ar (P.R. Fillottrani).
[1] See for example http://www.tipharma.com and http://www.phuse.eu.

http://dx.doi.org/10.1016/j.datak.2015.07.004
0169-023X/© 2015 Elsevier B.V. All rights reserved.

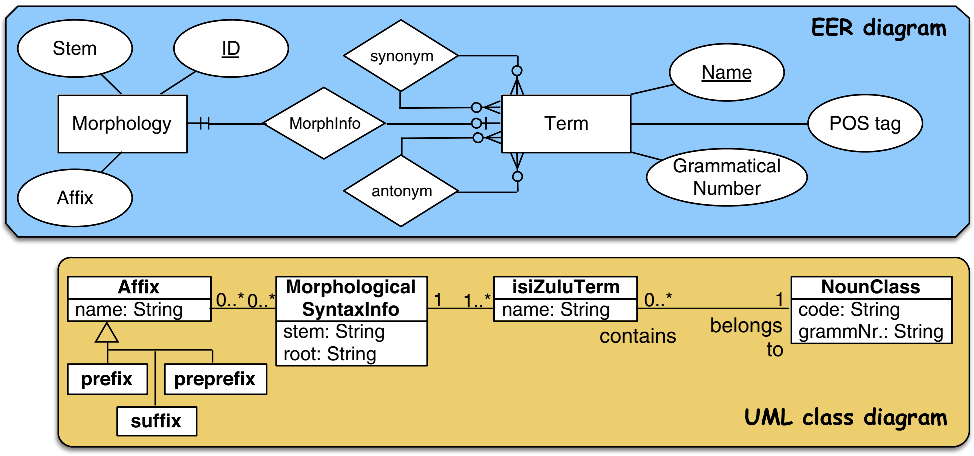

Fig. 1. Example of an integration scenario with two diagrams in different languages (names for the two left-most associations omitted due to width limitations).

termbank, which assumes it to be valid for all 11 official languages in South Africa. The UML diagram focuses on specific aspects of isiZulu terms for an isiZulu termbank that also requires some application layer processing due to its agglutinative characteristics. These diagrams are stylized and simplified from the actual model that is being developed for the isiZulu termbank at the University of KwaZulu-Natal [15], so as to illustrate the case rather than present models that span a few pages. For instance, to record *umuntu* (person/human) in the termbank, linguists need to have separated out the stem (*ntu*), the root (*nt*), suffix (*u*), prefix (*umu*) and pre-prefix (*u*), and besides grammatical number (singular/plural/mass) also the noun class (there are 17—*umuntu* belongs to class 1). This enables posing queries such as “retrieve all terms with root ‘nt’” or “retrieve all nouns of class 1a”, and computing various combinations in the business layer, as normally isiZulu terms are ordered alphabetical by stem, not full term, where the full term is computed using the affixes of the noun class. Going by terms in the two models, there clearly are similarities, but it is unclear how to link them such that the overall model and their associated data remains consistent. We return to this problem in Section 4.2, where the linking is assessed and resolved.

‘Traditional’ information systems development and management exhibit the capability of linking models represented in different languages only at the physical schema layer [16] or only for conceptual models represented in the same CDM language [17,18]. Some results are available on transformations between CDM languages (e.g., [19–21]), but they cover only a subset of all the languages' features, as subtle representational and expressive differences in the languages make this task very difficult. Consequently, current tools offer only very limited functionality in linking or importing models represented in one language into one represented in another language. For example, mandatory and disjointness is catered for, but weak entity types, identification, or attributes are not [16,20,22].

Any differences between the main CDM languages—UML Class Diagrams, ER, EER, ORM, and ORM2—may seem merely terminological, but it is known not to be the case either from a metamodeling viewpoint [23] or even within the same family of languages [24]. Also, what may seem very different at first glance may actually be the same, or at least share part of their meaning. This concerns possible agreements or differences in ontological foundation or commitment among the CDM languages. However, the state of the art in this area has focused primarily on a single CDM language and only for UML and ORM (e.g., [24,25]), and across languages only at the level of a few illustrations [26]. Put differently, it is unknown to what extent the languages agree on their underlying principles for modeling information. This is an obstacle for mapping and transformation algorithms in 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, from a cognitive viewpoint, a more precise insight in the commonalities and differences in the underlying modeling principles will contribute to the understanding of the extent to which the language features affects modeling information to tools, methods, and methodologies for CDM development and maintenance.

An approach toward solving these issues is to devise a comprehensive formalization of the languages so as to manage their interaction. To be able to do so, however, it first should be clear what entities and constraints exist in each language, how they can be used in the language, and how the differences can be reconciled without changing the languages. Put differently, not a comparison of the respective metamodels, but a *single integrated metamodel* inclusive of all language features is needed, so that one can unify the CDM languages and design transformation algorithms at the conceptual layer in software and database development. We designed such a unifying metamodel for the static, structural components and constraints of UML 2.4.1 Class Diagrams, EER, and ORM2/FBM, which is, to the best of our knowledge, the first of its kind. The metamodel is ontology-driven in the sense that our arguments and modeling decisions use motivations taken from the philosophical concept of Ontology, and ontologies in Computer and Information Sciences. At this stage, we are not concerned with the argument of convenience to fit with an a *priori* chosen logic language.

The unification makes clear the differences between and commonalities of the selected CDM languages. Notably regarding the entities in the languages, they agree only on Relationship (association) and Subsumption, Role (/association end/relationship