



Sanity-Checking Multiple Levels of Classification

A Formal Approach with a ConceptBase Implementation

Thomas Kühne¹  and Manfred A. Jeusfeld² 

¹ Victoria University of Wellington, Wellington, New Zealand
thomas.kuehne@vuw.ac.nz

² University of Skövde, IIT, Skövde, Sweden
manfred.jeusfeld@acm.org

Abstract. Multiple levels of classification naturally occur in many domains. Several multi-level modeling approaches account for this and a subset of them attempt to provide their users with sanity-checking mechanisms in order to guard them against conceptually ill-formed models. Historically, the respective multi-level well-formedness schemes have either been overly restrictive or too lax. Orthogonal Ontological Classification has been proposed as a foundation that combines the selectivity of strict schemes with the flexibility afforded by laxer schemes. In this paper, we present a formalization of Orthogonal Ontological Classification, which we empirically validated to demonstrate some of its hitherto only postulated claims using an implementation in CONCEPTBASE. We discuss both the formalization and the implementation, and report on the limitations we encountered.

Keywords: multi-level modeling · well-formedness · integrity constraints

1 Introduction

Modeling languages intended to support conceptual modeling differ to the extent by which they support modeling domains with multiple classification levels; specifically how explicitly they represent such domain classification within models. A long history of modeling mechanisms that attempt to support the modeling of multiple classification domain levels includes materialization [29] and power-types [27], both implying concepts that go beyond individuals and their types. Telos [25] pioneered support for an unbounded number of classification levels and DeepTelos [18] added support for *deep characterization* [6].

Unfortunately, having to manage more than two classification levels increases the potential of creating ill-formed models, i.e., models that cannot be given a sound interpretation. It has been argued that the complexity of contemporary conceptual modeling is akin to the complexity of programming large computer systems and therefore analog complexity management strategies are needed [14]. A well-known discipline within the area of multi-level modeling [8] for enforcing sound models is “strict metamodeling” [3], which is widely accepted to be highly selective, but has been equally widely criticized for being too inflexible [13,24,12].

Multi-dimensional multi-level modeling (MDM), based on the notion of “Orthogonal Ontological Classification” [21], has been proposed as a multi-level modeling paradigm that claims to enjoy the same selectivity as “strict meta-modeling” but without incurring the latter’s downside of requiring modelers to employ workarounds for several commonly occurring modeling scenarios. However, to this date, MDM has only been described informally, making it difficult to verify or validate its claims.

Since CONCEPTBASE had been successfully used to formalize the multi-level modeling approaches DDI [26], DeepTelos [18], and MLT* [16], we set out to

- develop a formalization of MDM,
- investigate whether CONCEPTBASE’ specification language is sufficiently expressive to support this formalization,
- examine CONCEPTBASE’s efficiency when supporting MDM, and
- empirically validate some of the MDM claims.

In this paper, we first further motivate the need for well-formedness checking of models featuring multiple levels of domain classification and then briefly compare two existing approaches to MDM [21] in Section 2. We subsequently present an MDM formalization using many-sorted first-order logic in Section 3 and follow with a description of an implementation of the formalization using CONCEPTBASE in Section 4. We finally, before concluding, discuss the formalization, its implementation, and lessons learned in Section 5.

2 Sanity Checking

Enforcing well-formedness requirements on models or programs is a well-established technique to ensure that the latter have a sound semantics. In particular, well-formedness requirements have been effectively used as preconditions to the analysis, interpretation, execution, etc., of models, protecting semantics implementations to trip over problematic structures such as circular or dangling references, to name just two of many sources for ill-formed scenarios.

Beyond serving this purpose, however, well-formedness constraints may also be used to alert users to structures that would not necessarily create problems for semantics implementations, but instead contain conceptual issues such as performing a category mistake. Multi-level models, in particular, provide a richer source for conceptual issues in user models, compared to traditional two-level counterparts. In general, such conceptual issues are harder to find than violations of straightforward structural requirements since they involve the semantics of concepts.

Providing respective solutions is becoming increasingly important due to the dependence of societies on reliable data and the significant amount of higher-order concepts naturally arising not only in specialized domains such as biology, or process metamodeling, but also in such commonplace domains as covered by UNICLASS classifications [28] and Wikidata [9,10]. Brasileiro et al. report that in 2016 Wikidata contained 6,963,059 elements involved in instantiation chains of lengths three [9].

2.1 Detecting Ill-Conceived Conceptualizations

Consider Figure 1 which shows a condensed version of a modeling scenario that was part of Wikidata in 2016 [9, Figs. 3 & 4]. The rightmost “instance-of” relationship can be derived from two Wikidata claims: First, that Tim Berners-Lee is a scientist and, second, that “Scientist” is a subtype of “Profession”. From these claims one can conclude that Tim Berners-Lee is a profession, which obviously does not make sense. Dadalto et al. observed that Wikidata no longer supports this particular incorrect inference, but that this is not a result of applying a general solution to eradicate all such issues. Equivalently ill-formed model fragments, e.g., a certain “Frank Hilker” being inferable as a “Position” are still pervasive in Wikidata, affecting many areas including biology, gastronomy, awards, professions, and sports [10]. Regarding the three “Anti-Patterns” Brasileiro et al. identified as characterizing ill-formed model fragments, they found that 15,177 Wikidata elements were involved in “Anti-Pattern 1”, and 7,082 were involved in “Anti-Pattern 3” [9].

In general, such nonsensical inferences cannot be mechanically detected without attaching semantics to the concepts involved and, based on those semantics, computing that a claim is made involving incompatible concepts.

Fortunately, however, nonsensical models like that in Figure 1 can still be mechanically detected without having to attach rich semantics to the concepts involved. For instance, by associating “order”-values to the concepts, e.g., by categorizing Tim Berners-Lee as an order-0 concept and Profession as an order-2 concept, it becomes apparent that the former cannot be a direct instance of the latter. Likewise, a specialization relationship between an order-1 concept Scientist and an order-2 concept Profession is equally unsound with respect to a set-theoretic interpretation of the model fragment.

Having to manually assign order values to each model element would be onerous, however even in the absence of such information, the scenario in Figure 1 can still be detected to make unsound claims based on its inconsistent relationships. The “instance-of” relationship between Scientist and Profession is necessarily incompatible with the simultaneous claim that the former is a subtype of the latter, regardless of the absolute order values associated to these concepts. There is an inherent contradiction in instantiation requiring the two orders to differ by one and specialization requiring that the two orders are identical.

The above explains the call for “Ontological Anti-Patterns” that can be used to detect such ill-formed scenarios [14,9]. In contrast, the approach underlying this paper was not arrived at by mining data for problematic patterns; rather the well-formedness constraints we are considering originated from the motivation to ensure that models have a sound set-theoretic interpretation.

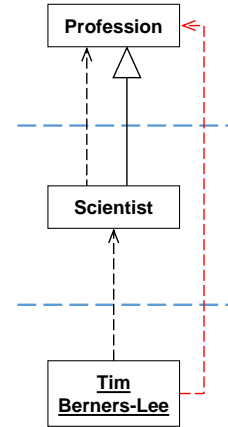


Fig. 1. Semantically Flawed Model

2.2 Previous Attempts

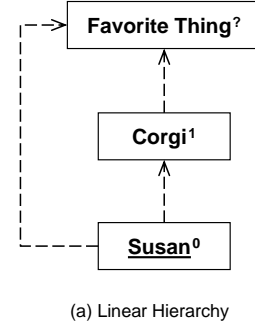
The original attempt to exclude ill-formed user models in the context of modeling with multiple levels of abstraction was “strict metamodeling” [3,4]. Based on a single principle, it rules out a huge class of conceptual errors, including those characterized by anti-patterns AP1–AP3 in [9]. The downside of its very conservative nature is that it also forbids users from adequately modeling a number of naturally occurring domain scenarios. These force users to employ workarounds that lead to “unnatural” solutions [23, section 8.1] or a duplication of elements [22], which not only add complexity of their own but also necessitate the introduction of additional constraints.

Many approaches aim to avoid the aforementioned downsides by using various concepts. The one most founded on ontological correctness is Almeida et al.’s MLT* which supports the adequate modeling of more demanding domain scenarios through the use of *orderless types* [2,11]. While a disciplined use of the approach retains sanity-checking abilities for a large proportion of a user model, the remaining part, involving orderless types, cannot be fully checked anymore. Some users may hence unintentionally exploit orderless types to create unsound models, thus undermining the rigor that MLT* otherwise supports.

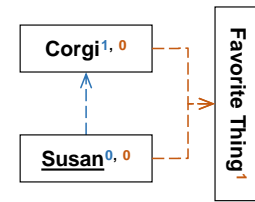
2.3 Orthogonal Ontological Classification

“Multi-Dimensional Modeling” (MDM) based on the notion of “Orthogonal Ontological Classification” claims to fully retain the sanity-checking capabilities of “strict metamodeling” without incurring its downsides, while avoiding to create loopholes that can be exploited [21]. It claims to retain the same rigor for local hierarchies, referred to as “classification clusters”, and argues that inter-cluster relationships cannot give rise to conceptually ill-formed models. It addresses challenging scenarios in which elements are ostensibly classified by multiple classifiers of different order (cf. Figure 2(a)), by maintaining that such overlapping classifications are best understood as occurring from different separate dimensions (cf. Figure 2(b)), using a “separation of concerns” approach.

Although MDM is inherently focused on precise well-formedness criteria for models and its description elaborates on a number of constraints to be enforced, the respective descriptions are informal and to date no complete publicly available implementation of the ideas has been available. We therefore set out to investigate the suitability of CONCEPTBASE for realizing an MDM implementation, both in terms of the expressiveness of its specification language and the efficiency of its optimized deductive database engine.



(a) Linear Hierarchy



(b) Multi-Dimension Hierarchy

Fig. 2. Traditional vs Orthogonal Classification

3 Formalization

Our formalization of MDM in `CONCEPTBASE` is based on `TELOS` (see Section 4), but for better accessibility we present a technology-independent formalization in this section. It not only covers a deliberately restrictive version of MDM, as outlined in [21, section 4.3], but also includes characterization potency [20]. We do not include well-formedness rules concerning element features (e.g., attributes), since our focus is on validating MDM’s main principles, which is possible while making very little reference to element features. In the following, due to space restrictions, we only present our formalization of the aforementioned MDM version without elaborating on the rationales behind the original constraints. However, wherever we deviated from the original, informally described, constraints, we state our motivation and reasoning.

The model structures we are concerned with are graphs over elements. The latter are sometimes referred to as “clabjects” [3], because they can play the role of a class or an object, or both at the same time. These elements are connected with relationships, of which we only cover classification and generalization here, as other relationships are not restricted by MDM. Since elements have potencies that belong to dimensions (cf. Figure 2) and relationships belong to dimensions as well, we use the many-sorted signature of equation 1.

We denote an element e with a potency p in dimension d as e^{p_d} . If e_2 classifies e_1 in dimension d , we use $e_1 :_d e_2$. If e_2 generalizes e_1 in dimension d , we use $e_1 \prec_d e_2$. The combination $e_1 : \prec_d e_2$ is used as a shortcut for representing that e_2 is either a classifier or a generalization of e_1 . We use a “+” superscript to denote sequences of relationships of at least length one. For instance, $e_s :^+ e_e$ represents the scenarios $e_s : e_e$, $e_s : e_1 : e_e$, $e_s : e_1 : e_2 : e_e$, etc.

In the following, we use labels for our well-formedness constraints that correspond to the labels C_1 - C_4 suggested in [21, section 4.3]. Since the latter do not cover characterization potency, we use a C_0 prefix for our respective constraints.

The first characterization potency constraint C_{0a} covers two aspects: First, upon instantiation potency must decrease, and second, only non-zero potency elements can be instantiated.

C_{0a} : *Instances must have a potency that is strictly lower than that of their classifiers and classifiers must have potencies greater than zero.*

$$\forall e_{1,2}, p_2, d : e_1 :_d e_2^{p_{2d}} \rightarrow \exists p_{1d} : e_1^{p_{1d}} \wedge 0 \leq p_{1d} < p_{2d} \quad (C_{0a})$$

Note the omission of any potency information on e_1 in the premiss. Underspecification with respect to element potency is allowed, however, we deemed it to be appropriate to enforce the specification of a potency value in case an element has a classifier with an explicit potency. We thus require all instance-classifier pairs where the classifier specifies a potency to obey the rules of characterization potency, not just those where both participants have explicit potencies.

The second characterization potency constraint C_{0b} requires that subtypes must not have a potency that is lower than the supertype potency.

C_{0b} : *Subtypes must have an equal or higher potency than their supertypes.*

$$\forall e_{1,2}, p_{1,2}, d: e_1^{p_{1d}} \prec e_2^{p_{2d}} \rightarrow p_{1d} \geq p_{2d} \quad (C_{0b})$$

Constraint C_1 , named *disjoint feature sets* and informally described in [21], is designed to avoid having to disambiguate access to element features in case multiple classifiers of an element define a feature with the same name. In the constraint definition below, the pattern $(e_1.a) : e_2$ represents, w.r.t. e_1 , the existence of a feature a with the type e_2 .

C_1 : *Elements that classify or generalize the same element, must not define features with the same name.*

$$\begin{aligned} \forall e_{0,1,2,3,4}, a_{1,2}: \\ (e_0 : e_1 \vee e_0 \prec^+ e_1) \wedge (e_0 : e_2 \vee e_0 \prec^+ e_2) \wedge \\ (e_1.a_1) : e_3 \wedge (e_2.a_2) : e_4 \wedge a_1 = a_2 \rightarrow e_1 = e_2 \end{aligned} \quad (C_1)$$

Even though we exclude multiple classification and multiple generalization within one dimension (via constraints C_{3a} & C_{3b}), we still need the above constraint to account for potential name clashes produced by multiple classification/generalizations from different dimensions.

Constraint C_2 , named *bottom-level overlapping* in [21], ensures that there is a unique dimension in which an element that is classified from multiple dimensions can be instantiated into. We cover this aspect with our constraint C_{2a} .

C_{2a} : *Elements with potencies in more than one dimension must not have more than one non-zero potency value.*

$$\forall e_0, p_{1,2}, d_{1,2}: e_0^{p_{1d_1}} \wedge e_0^{p_{2d_2}} \wedge d_1 \neq d_2 \wedge p_{1d_1} > 0 \rightarrow p_{2d_2} = 0 \quad (C_{2a})$$

Note that potency values of zero prevent instantiation (cf. constraint C_{0a}), and that we do not specify a classifier for e_0 since we want to allow for e_0 to be a top-level element with a manually assigned potency. We thus deviate from the “*bottom-level overlapping*” focus of the original C_2 constraint since we deemed that the constraint was in essence about preventing the potential of instantiation into more than one dimension, as opposed to only achieving this for elements that have explicit classifiers.

Since it is possible to omit potency specifications for the purposes of under-specification, an element could potentially be instantiated into multiple dimensions without constraint C_{2a} preventing such a scenario, since it (in combination with constraint C_{0a}) only covers cases featuring explicitly specified potencies. Constraint C_{2b} below addresses this by ensuring that instantiation may only occur into one dimension only even in the absence of any potency information.

C_{2b} : *All instantiations from a classifier must be into the same dimension.*

$$\forall e_{1,2,3}, d_{1,2}: e_1 :_{d_1} e_3 \wedge e_2 :_{d_2} e_3 \rightarrow d_1 = d_2 \quad (C_{2b})$$

Constraint C_{2c} below is not part of the original MDM well-formedness suggestions, however, we felt that an analog to constraint C_{2a} was needed that addressed the reception of type facets through specialization, thus complementing the classification focus of the original C_2 . Since constraint C_{2a} restricts elements to instantiation into one dimension only, it seemed inappropriate to allow elements to receive type facets from other dimensions via specialization.

C_{2c} : *Elements participating in multiple dimensions must not entertain generalization relationships in their potency-zero dimensions.*

$$\begin{aligned} & \forall e_{1,2}, d_{1,2} : \\ & \quad e_1 \prec_{d_1} e_2 \wedge \text{member}(e_1, d_2) \wedge \\ & \quad d_1 \neq d_2 \rightarrow \exists p_{d_1} : e_1^{p_{d_1}} \wedge p_{d_1} > 0 \end{aligned} \quad (C_{2c})$$

where $\text{member}(e, d) =$
 $\exists e_1 : (e :_d e_1 \vee e_1 :_d e \vee e \prec_d e_1 \vee e_1 \prec_d e)$

Note that the “member”-predicate does not require an element to have an explicit potency in a dimension. Dimension membership is solely acquired via respective relationships. This supports the underspecification of potency values, while simultaneously allowing checking for inappropriate type facet acquisition from dimensions that an element cannot be instantiated into anyhow.

The C_3 constraint, named *connected classification clusters* in [21] requires all elements within a dimension to form a single tree-shaped “classification cluster”. It prohibits disjoint clusters, containing `instanceOf` relationships, that declare the same dimension. Since the single cluster needs to be tree-shaped, we rule out multiple classification within a dimension with constraint C_{3a} .

C_{3a} : *Elements must not have more than one classifier within a dimension.*

$$\forall e_{0,1,2}, d : e_0 :_d e_1 \wedge e_0 :_d e_2 \rightarrow e_1 = e_2 \quad (C_{3a})$$

Note that ruling out multiple classification within a dimension does not represent nearly as much of a limitation as it would in an approach that did not support multiple classification from multiple dimensions.

Although the original C_3 formulation does not imply it, we also rule out multiple generalization (aka, multiple inheritance). It acknowledges that our implied language design currently does not support any merging mechanisms and/or semantics that a useful approach to multiple generalization should feature.

C_{3b} : *Elements must not have more than one supertype within a dimension.*

$$\forall e_{0,1,2}, d : e_0 \prec_d e_1 \wedge e_0 \prec_d e_2 \rightarrow e_1 = e_2 \quad (C_{3b})$$

We take a liberal approach to allowing multiple generalizations of different dimensions since they may be regarded as non-overlapping, i.e., do not require merging mechanisms.

The main “tree-shaped” aspect of the original C_3 constraint is taken care of by our constraint C_{3c} below.

C_{3c} : *Within a dimension, there must be only one classification cluster root.*

$$\forall d : (\exists e_3 : (\forall e_{1,2} : e_1 :_d e_2 \rightarrow e_1 :_d^+ e_3)) \quad (\text{C}_{3c})$$

It ensures that each dimension only has a single classification cluster, by ruling out classification forests that feature multiple roots.

The final original constraint C_4 , named *sound meta-hierarchies*, concerns general well-formedness requirements that would apply outside a multi-dimensional approach as well and correspond to, in spirit but not as restrictively, the regiment established by “strict metamodeling”.

The graphs implied by models must be free of cycles with respect to classification and generalization relationships.

C_{4a} : *The graph of instanceOf and specializationOf relations must be acyclic.*

$$\forall e : \neg(e : \prec^+ e) \quad (\text{C}_{4a})$$

Note that through the use of $:\prec^+$ we require every path with mixed classification and generalization relationships edges to be acyclic, as opposed to imposing the constraint only on pure classification and pure generalization paths respectively. Unlike the original C_4 constraint suggests, we do not restrict the context of the constraint to a single dimension only. In combination, these two choices lead to the rejection of a wider range of models with circular definitions, which we deem to be obviously ill-formed.

We do not need a constraint that establishes the *level-respecting*-property for classification hierarchies since a prerequisite for establishing respective ill-formed scenarios, is the ability of an element to be an instance of multiple classes in the same dimension and we rule out multiple classification scenarios via constraint C_{3a} .

An important component of the original C_4 constraint is that generalization relationships must not occur between elements of different order (i.e., of different set-theoretic classification power). We do not explicitly formalize element order but can infer when element orders must necessarily be different. If two elements are in the same classification branch, i.e., related to each other by one or more classification relationships, they must necessarily have different order values. Any such pair must not participate in the same generalization hierarchy, regardless of their relative positions in that hierarchy.

C_{4b} : *Elements in a classification path must not share a generalization hierarchy.*

$$\begin{aligned} \forall e_{1,2}, d : e_1 :_d^+ e_2 \rightarrow \neg \text{specConnected}(e_1, e_2, d) \quad (\text{C}_{4b}) \\ \text{where } \text{specConnected}(e_1, e_2, d) = \\ (e_1 \prec_d e_2 \vee e_2 \prec_d e_1) \vee \\ (\exists e_3 : (e_1 \prec_d e_3 \vee e_3 \prec_d e_1) \wedge \text{specConnected}(e_3, e_2, d)) \end{aligned}$$

Constraint C_{4b} could be generalized to cover all sources of order differences between elements but here we simply document what we were able to implement using CONCEPTBASE.

Before sharing our findings on the above eleven constraints in Section 5, we first present our respective CONCEPTBASE implementation.

4 Implementation

4.1 ConceptBase

CONCEPTBASE [15] is a deductive database system for managing models and metamodels. Its data model is based on the TELOS language [19] and its predicative specification language is based on Datalog with negation [1]. The latter uses a closed world assumption and guarantees terminating evaluations of –

- rules:** predicates that can infer information, similar to PROLOG predicates,
- constraints:** model integrity conditions which must always be satisfied, and
- queries:** supporting the identification of instances of custom query classes.

Around 30 rules and constraints in CONCEPTBASE define the TELOS semantics for instantiation, specialization, attribution, and relationships. TELOS is similar to the OMG’s MOF, in that TELOS can both be used to (in an extended variant or as is) directly represent user models, or to support the definition of modeling languages, which in turn are used to represent user models [7].

4.2 Realizing Multi-Dimensional Modeling with Telos

A fundamental design decision we had to make was to either build on TELOS’s definitions for instantiation and specialization, or to define a new language definition with custom instantiation and specialization relationships. We opted for the first alternative for the following reasons:

- MDM’s classification and generalization notions are compatible with TELOS,
- it minimized the effort, allowing us to focus on MDM-specific rules, and
- it allows a seamless adoption of MDM principles to TELOS.

Adding MDM well-formedness to TELOS well-formedness criteria can be achieved cleanly by employing CONCEPTBASE’s module system. A so-called `oHome` module, which defines relation semantics, provides the basis on which our `MultiDim` submodule builds on, to add potencies to elements, dimensions to relationships, rules, constraints, etc.

Early on in our experiments we learned that subjecting all TELOS objects to the MDM well-formedness principles resulted in undesirable performance. This is due to MDM’s inclusion of classification well-formedness and the fact that around 50% of predefined facts in CONCEPTBASE are classification-related. To address the lack of performance, we confined the application of MDM-specific constraints to MDM-specific elements by letting respective quantified variables in the constraints range over a custom-defined `Element` instead of the TELOS type `Individual` (cf. Section 5.2). `Element` represents the notion of a `Clabject`, i.e., a concept that can be an instance, a type, or both at the same time [3,5].

We use a total of 18 CONCEPTBASE rules to define the relations in Equation 5, e.g., `instanceOf/lab` and `specializationOf/lab`. Note the use of `lab` rather than `dim` which reflects the fact that the dimension properties attached to these

relationships are dimension *labels*. These can be user-defined and our validation scenarios include labels such as `Products`, `Favorites`, `Activities`, `Assets`, etc. These are labels of explicit dimension objects `Products`, `Favorites`, etc. which TELOS relationships link to (cf. Listing 1.1).

The “member” predicate used in constraint C_{2c} (see Section 3) is defined by two `mdrules`, one of which is shown in Listing 1.1, with the other one analogously taking care of specialization relationships.

```

1 $ forall inst/InstanceOf x/Element dim/Dimension
2   (inst dimension dim) and (From(inst, x) or To(inst, x))
3   ==> (x memberOf dim) $

```

Listing 1.1. Rule `mdrule1`

The TELOS class `InstanceOf` referenced in line 1 of Listing 1.1 classifies all explicit instantiation relationships. Likewise, `Dimension` classifies all dimension objects. With `(inst dimension dim)` we establish that the `inst` relationship is linked to a `dim` dimension object.

The next two premisses (line 2 of Listing 1.1) establish that element `x` participates in the instantiation relationship `inst`. From these it follows (in line 3) that element `x` is a member of dimension `dim`.

Of the eleven constraints we defined, we show our implementation of constraint C_{2c} in Listing 1.2, since it

- shows a usage of the custom-defined `memberOf` predicate (cf. Listing 1.1).
- is one of the richer constraints but still nicely demonstrates how readable `CONCEPTBASE` constraints are.
- exhibits the slight implementation inelegance of dealing with both dimension objects and dimension labels.

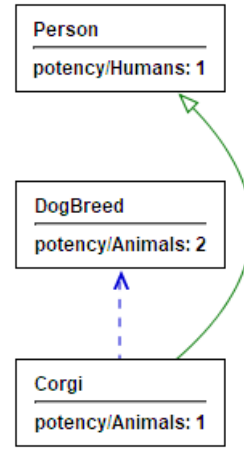


Fig. 3. C_{2c} -violating model

```

1 c2c: $ forall x,c/Element lab/Label dim/Dimension
2   (x specializationOf/lab c) and
3   (x memberOf dim) and not Label(dim,lab)
4   ==> exists p/Integer (x potency/lab p) and (p > 0) $

```

Listing 1.2. Constraint C_{2c}

Figure 3 shows a `CONCEPTBASE` screenshot of a model which is rejected due to violating constraint C_{2c} . Here, `Corgi` cannot claim to simultaneously be a classifier for both the `Animals` and the `Humans` dimensions.

The implementation of constraint C_{4b} in Section 3, shown in Listing 1.3, shows in line 2 how to use an operator like `:+` in `CONCEPTBASE`.

```

1 c4b: $ forall x,y/Element lab/Label
2   (x instanceOf_trans/lab y)
3   ==> not (x specConnected/lab y) $

```

Listing 1.3. Constraint C_{4b}

Line 2 in Listing 1.3 establishes that an element x is in the same instantiation branch as another element y , through any non-zero number of “instanceOf” relationships, in which case those two elements must not be in the same generalization hierarchy, i.e., must not be connected via any specialization relationships. The symmetric and transitive `specConnected` relationship is concisely defined by rules `mdrule17` & `mdrule18`, each rule fitting in one line.

We separated constraint- from rule definitions by using two separate Telos source files since it is often desirable to not enforce constraints, e.g., when developing models where intermediate editing states are not well-formed or when defining negative validation examples. We were thus able to include constraints only if and when we wanted to demonstrate that it passes or fails validation.

We implemented constraint C_{3c} as a CONCEPTBASE query rather than as a constraint since we wanted to avoid being forced to have all of our validation scenarios conform to constraint C_{3c} . A query allows one to check a model for a property on request, and unlike a constraint, can point out culprits in a visual manner. Figure 4 shows a CONCEPTBASE screenshot in which our query representing constraint C_{3c} was used to identify multiple classification cluster roots in a model. From a usability standpoint it can be argued that such visual support can be helpful compared to having to scan CONCEPTBASE error messages for the respective `Element` names.

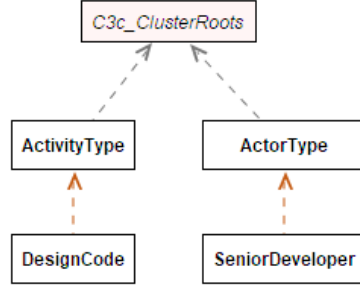


Fig. 4. Queries in CONCEPTBASE

4.3 ConceptBase Visualization Support

Beyond supporting the representation of MDM models and allowing them to be checked against well-formedness rules, we also implemented some visualization support. Note the colored relationships in the CONCEPTBASE screenshots (Figures 3 & 4). Modelers can specify arbitrary RGB colors when defining dimensions such as `Animals`. Element attributes and potencies are rendered below an `Element`’s name, instead of the standard CONCEPTBASE approach that

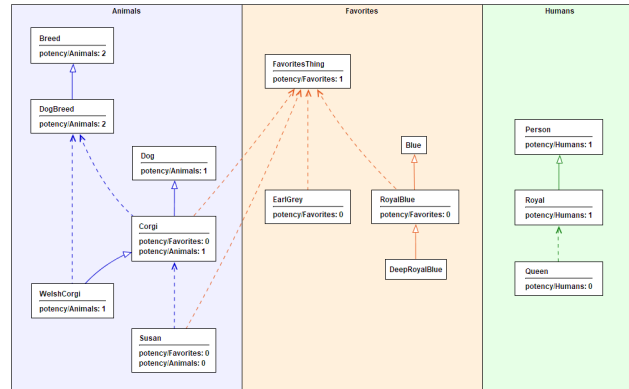


Fig. 5. Explicit dimensions in CONCEPTBASE

visualizes every attribute on its own and draws links between them and their owning elements.

Finally, we support the visualization of dimensions as such, using colored backgrounds, which are semantics-free but visually structure model content. Figure 5 shows a sample model featuring three dimensions. We omitted associations and links but note that these could have of course crossed dimension boundaries. We have made our CONCEPTBASE implementation source files and a set of models we used for validation purposes – in source format but also as PNG files – available at <https://conceptbase.sourceforge.net/mdm-er2023/> [17].

5 Discussion

In the following, we first discuss our formalization choices and the consequences resulting from them. In Section 5.2 we then discuss the merit of CONCEPTBASE as a supporting tool.

5.1 Formalization Discussion

The MDM paper our work is based on [21], proposes C_1 – C_4 “constraints” that informally describe one possible realization of the approach and are rather constraint categories, each often requiring multiple formal constraints to be covered, hence the use of our a , b , etc. sublabels. Whenever constraint definitions called for precision that was not in the informal descriptions or suggested generalizations, we often were able to improve or expand on the informal design: We

- generalized the exclusion of classifier feature clashes (C_1) to include super-type feature clashes (cf. constraint C_1).
- generalized the prohibition to instantiate a “bottom-level” element participating in multiple dimensions into more than one dimension (C_2), to include elements at any level. For instance, top-level elements may entertain potencies from multiple dimensions and should not be instantiatable into multiple dimensions either (cf. constraint C_{2a}).
- not only exclude cycles within classification- and generalization hierarchies respectively (C_4), but in general, i.e., cycles comprising mixed relationships of the former kinds, are excluded as well (cf. constraint C_{4a}).
- explicitly support dimension underspecification, i.e., allow dimensionless relationships and/or elements without explicit potencies.
- eschewed the notions of explicit “levels” and element-“order”, thus making our design agnostic to the explicit presence or absence of such notions. Instead, we inferred order differences from classification relationships.

Note that the latter choice elegantly targets the root cause of soundness violations and could be regarded as avoiding overspecification in comparison to a level-based approach. However, this design choice obviously means that we do not support manual “order” assignments or “level” allocations. Respective values are always inferred from relationships, i.e., we currently do not support any modeler-supplied claims about such values that could be checked for accuracy.

Generality Note that our `CONCEPTBASE` implementation does not cover all inferable order differences. Constraint C_{4b} identifies order-differences for elements in the same classification branch but, for instance, does not account for elements in different branches with known different path lengths to a shared root.

We still have to begin a quite involved investigation into whether `CONCEPTBASE`'s expressiveness is sufficient to infer a larger set of elements with order-differences. It is clear already, though, that the readability and execution efficiency of constraint C_{4b} would significantly suffer. An enhanced version of constraint C_{4b} would have to compare edge counts between different paths, which may be beyond what `CONCEPTBASE` can provide. The current syntax definitely does not support an extra edge-count parameter in addition to our `"/lab"` dimension label parameter.

For now, we are satisfied with the coverage our current version of constraint C_{4b} achieves for the following reasons:

- The constraint can be concisely formulated and is very readable.
- It does not require the use of `CONCEPTBASE` queries and/or functions which are more complex, i.e., require a much deeper skill set to develop.
- Whether `CONCEPTBASE` supports a better version is unclear at this stage.
- The discrimination power of our current constraint C_{4b} compares very favorably to approaches based on a small set of anti-patterns and is already optimal with respect to recognizing specialization connectivity.

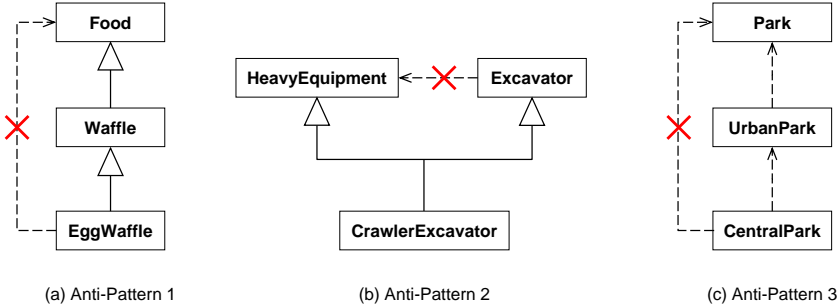


Fig. 6. Wikidata Anti-Pattern Scenarios (cf. [9, Fig. 6, Fig. 8, Fig. 10])

MDM Well-Formedness vs Anti-Patterns Although our current implementation does not infer all possible element order differences that could potentially give rise to unsound generalization hierarchies, it easily covers all scenarios detected by the Anti-Patterns AP1–AP3 defined in [9], plus many more. Figure 6 shows three Wikidata model fragments, which exemplify violations of three Anti-Patterns Brasileiro et al. used to detect ill-formed modeling scenarios. In general, Anti-Patterns represent schemata, i.e., will detect a wide range of unsound model fragments, not just very specific configurations. For instance, the generalization hierarchy involving `Food` and `EggWaffle` in Figure 6(a), could involve arbitrarily many generalization relationships; as long as a classification relationship between the bottom and top elements exists, the model is not sound.

Note that there is a single reason as to why AP1 & AP2 appropriately reject offending models: Elements, that are connected via generalization relationships, must have the same order; otherwise, no sound set-theoretic interpretation of the respective model exists. Our constraint C_{4b} simultaneously covers AP1 & AP2 since it targets the root cause that underlies the validity of these two anti-patterns. Unlike AP1, for instance, constraint C_{4b} also correctly rejects models like that in Figure 6(a) where the classification relationship is reversed, i.e., where Food is declared to be an instance of EggWaffle.

Our constraint C_{3a} takes care of AP3 violations, again, in a very general manner, i.e., the constraint is (trivially, in this case) agnostic to the number of classification edges involved. A less trivial detection, that does not simply rule out multiple classification, would have required the implementation of the “*level-respecting*” property mentioned in [21]. Since this property would require the comparison of classification path lengths, it belongs to the same “unclarified” category as the extension of constraint C_{4b} (see above).

We observed that anti-pattern scenario variations we considered in our validation sometimes violated more than one constraint. This appears to testify to an increased robustness of a sanity-checking approach that covers multiple soundness principles. For instance, the aforementioned variation of AP1 in Figure 6(a), violates both constraint C_{4a} & C_{4b} .

We acknowledge that the Anti-Patterns used in [9,10] were solely used as queries to search for ill-formed model fragments and hence should not be judged as integrity specifications. However, we note that for the purposes of ensuring the well-formedness of models, an approach based on constraints that embody fundamental soundness principles seems more suited than a collection of schemata that were devised on the basis of found integrity violations.

Overall, we did not attempt to create a full-fledged language design that resolves all possible design choices, e.g., we did not impose limitations on generalizations into multiple dimensions even though the respective semantics are undefined at this stage. We only attempted, and almost entirely succeeded, to capture the constraint categories C_1 – C_4 . CONCEPTBASE might support a full coverage but ascertaining whether that is the case is less than trivial and any respective measures would affect constraint readability and execution efficiency.

5.2 Implementation Discussion

Expressiveness It is remarkable how close CONCEPTBASE constraint implementations such as Listing 1.3 are to a concise logic formulation (cf. constraint C_{4b}). There is some contamination due to the need to distinguish between dimension objects and their corresponding labels but overall the CONCEPTBASE constraints are very readable and very well supported experimentation with variants.

As mentioned in Section 5.1, we did not implement the “*level-respecting*” property of C4 (addressing it trivially by excluding multiple classification) and our constraint C_{4b} is not as general as it theoretically could be. We plan to investigate whether there is a real hard limitation of CONCEPTBASE expressiveness or whether a rather involved implementation may be feasible after all.

Efficiency CONCEPTBASE’s evaluation of rules and constraints is not sufficiently performant to support all MDM constraints at the TELOS level. We therefore had to let our constraints range over a dedicated `Element` class instead of the TELOS class `Individual`, and let one rule range over `DimensionLabel` instead of `Label`, to achieve typical evaluation times of less than one second per instance of `Element`. Given those domain restrictions, our validation scenarios are checked very swiftly, in particular when incremental changes to existing models are made.

Usability Our emphasis was on exploring the feasibility of formalizing MDM constraints using CONCEPTBASE and we therefore paid little attention to usability concerns. For instance, we could have formulated all constraints as queries, which would have allowed them to be evaluated on demand only – thus decreasing model loading times and supporting intermediate invalid models – and produce visual pointers to offending elements (cf. Figure 4). However, the readability of our implementation would have suffered as a consequence. As an exception, we implemented constraint C_{3c} as a query because enforcing it on all models seemed limiting to users while the benefits of enforcing this particular integrity condition were not obvious to us.

We also considered implementing part of constraint C_{0a} as a query, giving users the chance to omit potency values and only providing them with a warning mechanism in case the underspecification is unintended (cf. Section 3). However, we eschewed the formulation of an additional query as we deemed the enforcement of the constraint to be appropriate.

Overall, we treated CONCEPTBASE mainly as a user model storage backend as opposed to an environment with optimal support for interactive modeling. We support visual presentation of dimensions (cf. Figure 5) but so far did not pursue further visualization support, e.g., rendering potencies as superscript values.

Utility for Formalization As expected, CONCEPTBASE proved to be invaluable for validating our formalization. In many cases, subjecting select modeling scenarios to our constraints simply confirmed the latter’s adequacy and/or the claimed properties of MDM. In some cases, however, CONCEPTBASE supported experimenting with variants, e.g., to explore alternative formulations or achieve better evaluation efficiency. By defining a validation suite of model scenarios that target all constraints respectively, we were able to trial tweaks and either confirm or disprove that they were still reporting ill-formed models and not reporting sound models and/or measure respective evaluation times.

Note that our suggested amendment to the original C_2 formulation in the form of a slightly wider constraint C_{2a} definition was arrived at during such constraint validation experiments. While working with respective validation scenarios, it seemed odd to forbid the clashing of multiple potency values greater than zero only for elements that have explicit classifiers (cf. Section 3).

Our regression validation suite (not including further scenarios that we used in general development) and a log of evaluating the respective models is part of the data we provided at [17].

6 Conclusion

The more critical the reliance on the conceptual integrity of a model is, the higher the need to eliminate avoidable conceptual mistakes. It is concerning that modeling concepts of societal importance, such as “gene”, “protein”, and “disease” are used inconsistently in models [10, Table 1]. Ontological sanity-checking of models is not a novel concept, but for multi-level models, it has in the past translated to users either needing to complicate their models as a result of having to work around overly strict well-formedness requirements, or users being subjected to loopholes that they may inadvertently exploit with ill consequences.

In this paper, we presented the first formalization of an approach [21] that reliably and independently of the modeling domain prevents a large class of ill-conceived conceptualizations without requiring modelers to explicitly provide semantic descriptions of the concepts they are using, or forcing them to work around unnecessary limitations imposed by overly strict well-formedness criteria when modeling naturally occurring domain scenarios. Our formalization does not rely on explicit “order” or “level” constructs, making it widely applicable, i.e., a candidate for adoption by other multi-level modeling approaches. While an implementation challenge has so far prevented us from realizing the full discrimination potential of the MDM paradigm, our implementation is very faithful to MDM principles and has more discriminative power than the anti-patterns in [9,10]. Being based on fundamental soundness principles, rather than attempting to match ill-formed model fragments, our implementation is not dependent on a comprehensive schematic capturing of such fragments.

We have empirically validated MDM claims and our implementation by using numerous sample models of which the most essential form a regression validation test suite that we resorted to any time we explored a constraint variant, e.g., to increase evaluation efficiency. In many cases we additionally created a systematic exploration of scenarios, in which, for instance, all combinations of relationship directions in specific scenarios were explored. Our slight modification of constraint C_{2a} and our introduction of constraint C_{2c} were the direct result of following such a tool-supported, scenario-based exploration approach.

We demonstrated that CONCEPTBASE is capable of supporting a concise, intuitive and sufficiently performant implementation of MDM well-formedness principles that did not require any coding at any stage. Even without prioritizing usability, we achieved decent notation support, including the rendering of UML-like attributes, colored relationships, and explicit dimension containers.

Despite that fact that future work remains with respect to exploring the expressiveness of CONCEPTBASE, which could potentially improve the coverage of constraint C_{4b} , we are convinced that our work is a suitable foundation for a further exploration of the MDM paradigm, allowing richer variants – such as supporting instantiation into multiple dimensions from a single element – to be considered and validated. Our formalization and public implementation open up these avenues not only for us, but also to any other researchers who may want to extend or adopt the approach to fit their frameworks.

References

1. Abiteboul, S., Hull, R.: Data functions, Datalog and negation. *SIGMOD Rec.* **17**(3), 143–153 (jun 1988). <https://doi.org/10.1145/971701.50218>
2. Almeida, J.P.A., Fonseca, C.M., Carvalho, V.A.: Comprehensive formal theory for multi-level conceptual modeling. In: *Proceedings of 36th International Conference on Conceptual Modeling*, vol. LNCS 10650. Springer (2017)
3. Atkinson, C.: Meta-modeling for distributed object environments. In: *Enterprise Distributed Object Computing*. pp. 90–101. IEEE (Oct 1997)
4. Atkinson, C., Gerbig, R.: Melanie: Multi-level modeling and ontology engineering environment. In: *Proceedings of Modeling Wizards’12*. ACM (2012)
5. Atkinson, C., Kühne, T.: The essence of multilevel metamodeling. In: *Proceedings of the 4th International Conference on the UML 2000*, Toronto, Canada. pp. 19–33. LNCS 2185, Springer Verlag (Oct 2001). https://doi.org/10.1007/3-540-45441-1_3
6. Atkinson, C., Kühne, T.: Rearchitecting the UML infrastructure. *ACM Transactions on Modeling and Computer Simulation* **12**(4), 290–321 (Oct 2003)
7. Atkinson, C., Kühne, T.: Concepts for comparing modeling tool architectures. In: Briand, L. (ed.) *Proceedings of the ACM/IEEE 8th MODELS*. pp. 398–413. Springer Verlag (2005)
8. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. *Software and Systems Modeling* **7**(3), 345–359 (2008). <https://doi.org/10.1007/s10270-007-0061-0>
9. Brasileiro, F., Almeida, J.P.A., Carvalho, V.A., Guizzardi, G.: Applying a multi-level modeling theory to assess taxonomic hierarchies in Wikidata. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. pp. 975–980. WWW ’16 Companion, International World Wide Web Conferences Steering Committee (2016). <https://doi.org/10.1145/2872518.2891117>
10. Dadalto, A.A., Almeida, J.P.A., Fonseca, C.M., Guizzardi, G.: Type or individual? Evidence of large-scale conceptual disarray in Wikidata. In: *Proceedings of Conceptual Modeling - 40th International Conference, ER 2021*. LNCS, vol. 13011, pp. 367–377. Springer (Oct 2021). https://doi.org/10.1007/978-3-030-89022-3_29
11. Fonseca, C.M., Almeida, J.P.A., Guizzardi, G., Carvalho, V.A.: Multi-level conceptual modeling: From a formal theory to a well-founded language. In: *Proceedings of the 37th International Conference on Conceptual Modeling (ER 2018)*. LNCS 11157, Springer Verlag (10 2018)
12. Frank, U.: Multilevel modeling - toward a new paradigm of conceptual modeling and information systems design. *Business & Information Systems Engineering* **6**(6), 319–337 (2014). <https://doi.org/10.1007/s12599-014-0350-4>
13. Gitzel, R., Merz, M.: How a relaxation of the strictness definition can benefit MDD approaches with meta model hierarchies. In: *Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics & Informatics*. vol. IV, pp. 62–67 (July 2004)
14. Guizzardi, G.: It’s patterns all the way down: Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. *ACM Lecture* (2020), <https://speakers.acm.org/lectures/13930>
15. Jeusfeld, M.A.: Metamodeling and method engineering with ConceptBase. In: *Metamodeling for Method Engineering*, pp. 89–168. MIT Press (2009)
16. Jeusfeld, M.A., Almeida, J.P.A., Carvalho, V.A., Fonseca, C.M., Neumayr, B.: Deductive reconstruction of MLT* for multi-level modeling. In: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. MODELS ’20* (2020). <https://doi.org/10.1145/3417990.3421410>

17. Jeusfeld, M.A., Kühne, T.: ConceptBase implementation of MDM. Project Web Site (2023), <https://conceptbase.sourceforge.net/mdm-er2023/>
18. Jeusfeld, M.A., Neumayr, B.: DeepTelos: Multi-level modeling with most general instances. In: Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016. pp. 198–211 (2016). https://doi.org/10.1007/978-3-319-46397-1_15
19. Koubarakis, M., Borgida, A., Constantopoulos, P., Doerr, M., Jarke, M., Jeusfeld, M.A., Mylopoulos, J., Plexousakis, D.: A retrospective on Telos as a metamodeling language for requirements engineering. *Requir. Eng.* **26**(1), 1–23 (2021). <https://doi.org/10.1007/s00766-020-00329-x>
20. Kühne, T.: Exploring potency. In: ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems (MODELS '18). ACM (2018). <https://doi.org/10.1145/3239372.3239411>
21. Kühne, T.: Multi-dimensional multi-level modeling. *Software and Systems Modeling* **21**(2), 543–559 (2022). <https://doi.org/10.1007/s10270-021-00951-5>
22. Kühne, T., Lange, A.: Melanee and DLM: A contribution to the MULTI collaborative comparison challenge. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. p. 434–443. MODELS '22, ACM, NY, USA (2022). <https://doi.org/10.1145/3550356.3561571>
23. Lange, A., Atkinson, C.: Multi-level modeling with LML – A contribution to the multi-level process challenge. *International Journal of Conceptual Modeling* **17** (Jun 2022). <https://doi.org/https://doi.org/10.18417/emisa.17.6>, special Issue: Multi-Level Process Challenge
24. de Lara, J., Guerra, E., Cobos, R., Moreno-Llorena, J.: Extending deep meta-modelling for practical model-driven engineering. *The Computer Journal* **57**(1), 36–58 (2012). <https://doi.org/10.1093/comjnl/bxs144>
25. Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M.: Telos: Representing knowledge about information systems. *Information Systems* **8**(4), 325–362 (1990)
26. Neumayr, B., Jeusfeld, M.A., Schrefl, M., Schütz, C.: Dual deep instantiation and its ConceptBase implementation. In: Proceedings Advanced Information Systems Engineering CAiSE 2014. pp. 503–517. Springer Int. Publ., Cham (2014)
27. Partridge, C., de Cesare, S., Mitchell, A., Odell, J.: Formalization of the classification pattern: survey of classification modeling in information systems engineering. *Software & Systems Modeling* **17**(1), 167–203 (2 2018). <https://doi.org/10.1007/s10270-016-0521-5>
28. Partridge, C., Mitchell, A., da Silva, M., Soto, O.X., West, M., Khan, M., de Cesare, S.: Implicit requirements for ontological multi-level types in the uniclass classification. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. MODELS '20 (2020). <https://doi.org/10.1145/3417990.3421414>
29. Pirotte, A., Zimányi, E., Massart, D., Yakusheva, T.: Materialization: A powerful and ubiquitous abstraction pattern. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94). pp. 630–641. Morgan Kaufman (1994)