Notes on the use of the powertype pattern*

André M. Demori[1,*,†], Julio Cesar Cardoso Tesolin[2,†] and Maria Cláudia Reis Cavalcanti[3,†]

[1]Military Institute of Engineering, Praça Gen. Tibúrcio, 80 - Urca, Rio de Janeiro - RJ, 22290-270

Abstract

Identifying the need for multi-level modeling patterns in conceptual models is a non-trivial task, especially when such needs are implicit or are heavily context dependent. This paper proposes some guidelines for identifying the use of the Powertype pattern, grounded in the Multi-Level Theory (MLT), using domain-independent competency questions to support this identification. The approach is illustrated through a case study that requires multi-level representations for entities and relationships at the type level. Our proposal aids the detection of hidden structures in models, contributing both to the construction of multi-level conceptual models and of ontology-driven conceptual models.

Keywords

modeling patterns, Powertype pattern, multi-level modeling, multi-level theory

1. Introduction

During the process of building a conceptual model, some entities present aspects and behaviors that lead to representing them as both class and an individual. While the former presents predicative characteristics, the latter presents specific characteristics. Fonseca et al. [1] state that this phenomenon occurs mainly when classes are also subject to classification, commonly seen in biological taxonomies.

This conceptual modeling challenge can be dealt with by using multi-level modeling patterns. A well-known multi-level pattern is the *Powertype* pattern [2, 3], which occurs when instances of one type (powertype) are specializations of another type (basetype). However, recognizing the need to use this modeling pattern is not straightforward. It requires the modeler to identify hidden entities and relationships that are not immediately apparent when designing the first versions of a conceptual model.

This work proposes a set of steps (guidelines) to support the modeler in the use of multi-level modeling patterns and how they can improve the resulting conceptual model. We analyze specific situations where the *Powertype* pattern may be applied and the benefits it brings. Additionally, for each situation, we formulate useful domain-independent competency questions that cut across modeling levels, which would otherwise remain unanswered. Furthermore, uncovering implicit model entities and relationships, discovering the possible patterns applicable to provide semantic enrichment helps in restoring the ontological commitment. In this manner, we consider these guidelines a relevant contribution of this paper to the process of ontological analysis of conceptual models because of its didactic nature on showing the multi-level structures potential to enrich representations. Finally, to illustrate the use of the proposed guidelines, a case study is presented, in which we revise the conceptual modeling of an existing ontology by applying the *Powertype* pattern[1]. Additionally, we show how the punning technique can be used to implement the identified *powertypes*.

This article is structured as follows: Section 2 provides the theoretical background on multi-level

Proceedings of the 18th Seminar on Ontology Research in Brazil (ONTOBRAS 2025) and 9th Doctoral and Masters Consortium on Ontologies (WTDO 2025), São José dos Campos (SP), Brazil, September 29 – October 02, 2025.

*Corresponding author.

†These authors contributed equally.

andrademori@ime.eb.br (A. M. Demori); jcctesolin@ime.eb.br (J. C. C. Tesolin); yoko@ime.eb.br (M. C. R. Cavalcanti)

0000-0002-0533-3395 (A. M. Demori); 0000-0002-0240-4506 (J. C. C. Tesolin); 0000-0003-4965-9941 (M. C. R. Cavalcanti)

[1]We use *Powertype* with a capital letter, followed by pattern, when referring to the Powertype pattern. We use *powertype* with a lowercase letter, when referring to the conceptual element itself (a type of type), used to denote classes that classify other classes.

conceptual modeling and foundational ontologies. Section 3 reviews the main related works that address multi-level modeling and the *Powertype* pattern. Section 4 presents the core contributions of this work. Section 5 illustrates the proposed approach through a case study, demonstrating both the revised modeling of an ontology and its operationalization. Finally, Section 6 summarizes the main findings and outlines directions for future research.

2. Background

Foundational ontologies such as UFO [4], GFO [5], BFO, and DOLCE [6, 7] provide formal semantics that are crucial for conceptual modeling. They enable the expression of both explicit and implicit characteristics of entities in a domain, supporting semantic enrichment and a strongly axiomatized representation. Leveraging these semantics requires an ontological analysis to uncover the metaphysical and structural foundations of the entities and their interrelations.

Ontology-driven conceptual modeling (ODCM) involves systematically analyzing a domain—often starting from pre-existing structures such as database schemas or ER diagrams—to reveal its underlying ontological commitments, identifying and addressing modeling patterns [8], including multi-level patterns [9] and *truthmakers* [10]. It is a critical challenge, as these are often absent or only implicitly represented in existing models. Reification is one of the strategies that could be useful in this process, once it brings to light hidden entities, promoting them to first-class citizens [10].

2.1. Multi-Level Conceptual Modeling

According to [11], an intension[2] characterizes each type. This intension identifies whether the type applies to a particular entity. Thus, if the intension of a type T_1 applies to an entity e, then e is said to be an instance of T_1. However, there are modeling cases where an intension of a type T_p applies to type T_1, i.e., type T_1 is an instance of a type T_p. This chain is illustrated in Figure 1(a). The *Powertype* pattern in conceptual modeling appears alongside the occurrence of this chain of instantiations, as shown in 1(b).

MLT incorporates two notions of powertype, one based on Cardelli's [3] definition and another based on Odell's [2]. According to Cardelli [3], this pattern is defined as follows: if T_b is a type, then T_p is the type whose instances are all the subtypes of T_b; if T_1 is an instance of T_p then T_1 is a subtype of T_b. T_p is referred to as *powertype(T_b)*, and T_b is referred to as *basetype*. On the other hand, Odell considers that there are specializations of the base type that may not be an instance of a *powertype*. MLT shows how both definitions are related to each other and how they can be used in different manners [11].

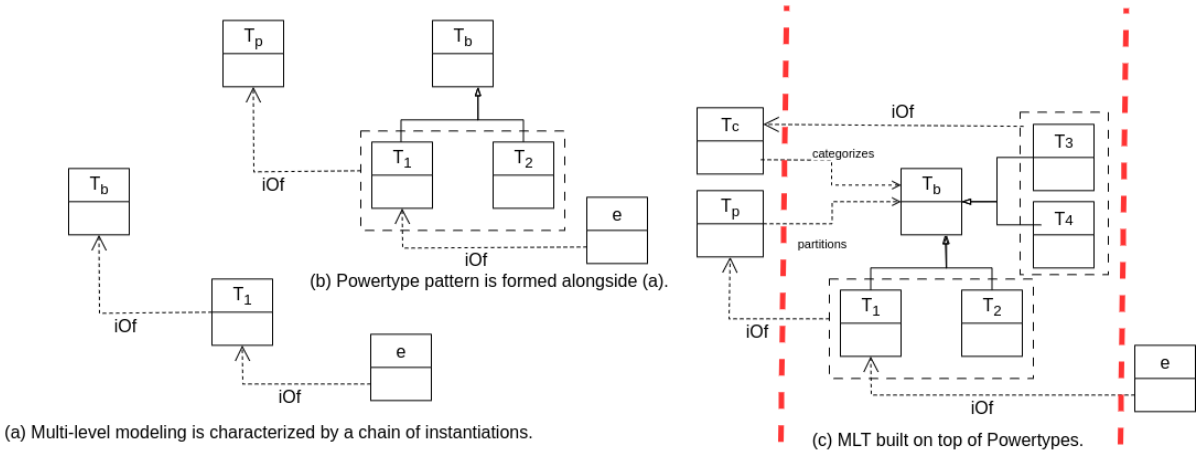

Figure 1: Multi-level modeling evolution.

[2]In this work we use the word intension meaning the set of characteristics or properties by which the referent or referents of a given word are determined. https://www.collinsdictionary.com/dictionary/english/intension (accessed September 9, 2025).

To create a well-founded theory about multi-level modeling, Carvalho et al. developed the Multi-Level Theory (MLT) [11, 12, 13, 14], which primarily characterizes the orders of types and the partitioning and categorization relations between types. Figure 1(c) summarizes in a visual way, the contribution of MLT to the *Powertype* pattern.

By the theory's definition, types that have *Individuals* as instances are classified as first-order types (1stOT), and types whose instances are first-order types are classified as second-order types (2ndOT), and so on, thus creating chains of instantiations. These are the basic types of Multi-Level Theory. Once the *Powertype* pattern has been recognized, it is necessary to check whether there is a partitioning or a categorization relationship between *powertype* and *basetype*. According to MLT, a *powertype* partitions the *basetype* if and only if each instance of the *basetype* is an instance of exactly one instance of the *powertype*.

Also, according to MLT [11], the categorization relation occurs between a *powertype* and a *basetype* when the intension of the *powertype* defines that its instances specialize the *basetype* according to a specific classification criterion. A variation of the categorization relation is called Complete categorization, in which the classification criteria defined by the intension of the *powertype* guarantees that each instance of the *basetype* is an instance of at least one instance of the *powertype*. A third variation is called Disjoint categorization, in which each instance of the *basetype* is an instance of at most one instance of the *powertype*.

3. Related Works

The academic literature has been proposing several techniques and theories to deal with multi-level modeling. Neumayr et al. [15] addressed the issue of multiple levels in conceptual modeling through what is called Multiple Level Objects (m-objects) and Multiple Level Relationships (m-relationships), which provide a representation with multiple levels of abstraction, encapsulating the different levels that relate to a single domain concept and integrating aspects of the different semantic abstraction hierarchies (aggregation, generalization and classification) into a single concretization hierarchy. However, as Carvalho et al. [13] noted in their work, this abstraction of levels can render the modeler unable to express whether instances of a higher-order type are disjoint and/or comprehensive types and also unable to determine the meta-properties.

Guizzardi and Almeida discuss stability patterns in conceptual models [16]. Then, the authors analyze several stability patterns, including multi-level modeling and reification of intrinsic and relational aspects. The authors provide an interesting explanation of how reifying intrinsic aspects in quality spaces and relations leads to more stable conceptual models. Moreover, the higher-order types can help to create conceptual models that focus on invariant aspects, turning a less rigid conceptual model.

In turn, Lara, Guerra, and Cuadrado propose a deep explanation about when and how to use multi-level modeling [17], discussing situations where the use of multi-level modeling is beneficial for the representation. The authors also introduce different techniques to work with multi-level and also address discussions about multi-level patterns, including the Powertype pattern. Halpin also presents a didactic work to describe some challenges in modeling subtypes [18], examining several subtype issues such as derivation options, subtype rigidity, and subtype migration. However, the author focuses on ORM technique to explore these concepts and not on a well-founded theory of a foundational ontology.

Guizzardi et al. [9] proposed a study towards an ontological analysis of powertypes, analyzing issues such as (i) powertype instances as universals, (ii) powertype instances as mereological sums (iii) powertype instances as variable embodiments. Also, the authors discussed the identity of instances and the classification relation (isClassifiedBy).

Despite these works offering similar approaches to address or analyze multi-level issues in conceptual modeling, they do not develop domain-independent competence questions, which helps reinforce that this pattern should be applied. Moreover, this paper proposes a didactic approach for working with multi-level in conceptual models, which can be considered a complement to related works.

4. Multi-Level Modeling Guidelines

As already mentioned, the problem we want to address is the difficulty in applying multi-level modeling patterns while modeling a domain. In this direction, this section presents some guidelines for applying multi-level modeling, assuming that it may bring benefits. Section 4.1 introduces the first steps based on the use of the reification technique. Section 4.2 focuses on identifying the appropriate level of attributes in a multi-level model. Finally, Section 4.3 takes us to additional steps where we deal with related hierarchical restrictions on multiple levels.

4.1. Multi-Level Identification First Steps

As mentioned in subsection 2.1, a typical multi-level modeling case is when a chain of instantiations occurs, i.e., when an entity e is an instance of T_1, which in turn is an instance of T_p. If we can evolve into a richer multi-level modeling, like the one in Figure 1(c), then it becomes possible to answer the following set of Competency Questions (CQ):

CQ1: Which types categorize T_b?
CQ2: Which types partition T_b?
CQ3: In which subtypes can one classify (categorize) an instance of T_b according to T_c?
CQ4: In which subtypes can one partition instances of T_b according to T_p?

However, it is not an easy task to identify this pattern and also to predict such CQs. It is a usual modeling practice to include types that have attributes or relationships that classify their instances. For example, consider the following two domain models: one that has a type T_b, which is classified by an $attr_1$ (Figure 2(a)), and another one where type T_b is classified by another type T_p (Figure 2(b)). In both cases, the above competency questions could not be answered. In the first case (Figure 2(a)), it is not clear what the meaning of $attr_1$ to T_b is. Similarly, in the second case (Figure 2(b)), the meaning of the "isClassifiedBy" relationship is unclear, as it does not distinguish whether the classification partitions or categorizes T_b, unless the cardinality of the relationship is explicitly defined. In other words, there are hidden entities and relationships that should come to light.

Guarino et al. [10] use reification in conceptual modeling to identify *truth-making* patterns. It involves making explicit entities while modeling a domain that would otherwise be implicit. Similarly, we examine the use of reification to make explicit the *Powertype* pattern. Thus, inspired by Guarino et al., we defined a set of steps that depart from reifying attributes and evolve the model up to the point that it forms a multi-level pattern. After applying these steps, it becomes possible to answer the previously defined CQ. It is worth mentioning that these steps are one of the possible methods to obtain a rich multi-level modeling.

Figure 2 illustrates the first steps to identify the *Powertype* pattern. It begins with a simple type T_b and its attribute $attr_1$ (Figure 2(a)). If $attr_1$ is a classifying attribute for T_b that is essential for its characterization, and which needs to have its own attributes, then it must be reified into a type T_p. Thus, in **Step 1**, the model evolves to Figure 2(b), where $attr_1$ corresponds to T_p, which is connected to T_b through the *isClassifiedBy* relationship.

Next, in **Step 2**, we need to check if there are specializations of type T_b that need to be represented. For instance, there might be other relationships departing from T_b that indicate that a subset of it may be related to a T_x, as shown in Figure 2(c). This indicates that a specialization of T_b may be hidden and must be revealed [19]. In this case, the model evolves to Figure 2(d), where a *subtype* set is created for type T_b.

Then, in **Step 3**, for each revealed specialization, we check if the subtype extension matches the extension of type T_p. If this is the case, it means that the *Powertype* pattern is formed, as shown in Figure 2**(e)**, where a dotted line represents the *instance Of* (iOf) relationship between type T_p(*powertype*) and the *subtypes* (T_1 and T_2). In addition, the relationship between the *basetype* and the *powertype* in this case is (*partitions*), which means T_b is completely partitioned into disjoint subtypes according to T_p.

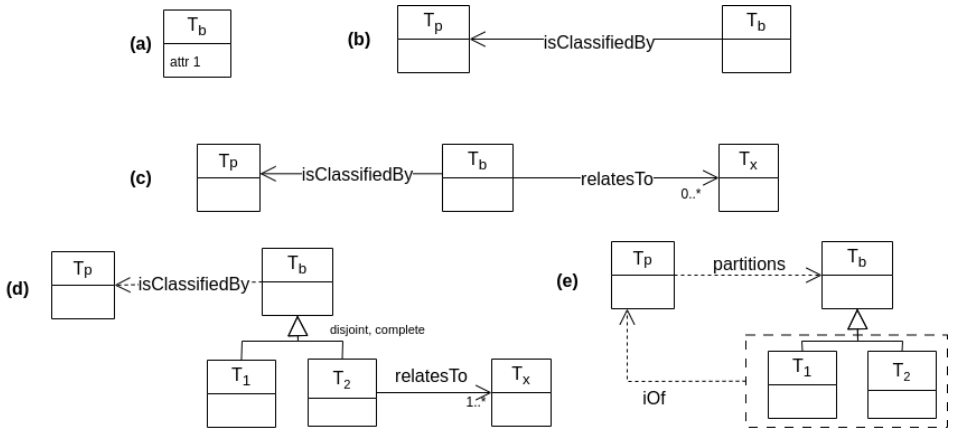

Figure 2: Multi-level identification first steps.

To illustrate the steps described above, Figure 3 presents a domain-specific example, where the *Powertype* pattern is formed having as its *basetype* the type Person. Initially, type Person has been modeled as having two classifying attributes, namely, *Employee Position* and *Academic Role* (Figure 3(a)). An instance of type Person, "Maria", would be represented with values "Coordinator" and "Professor", respectively, assigned to those attributes. After applying the steps (Figure 3(b)-(e)), the type Person becomes a *basetype*, which is categorized by the *Academic Role* type, partitioned by the *Employee Position* type, and specializes in two *subtypes*. Based on this example, we can formulate and answer the previously defined CQs, as follows:

- Which types categorize *Person*? Academic Role
- Which types partition *Person*? Employee Position
- In which Subtypes can I classify (categorize) an instance of *Person* according to *Academic Role*? Student and Professor
- Which *subtypes* of *Person* are instances of Employee Position? Manager, Coordinator, and Analyst.

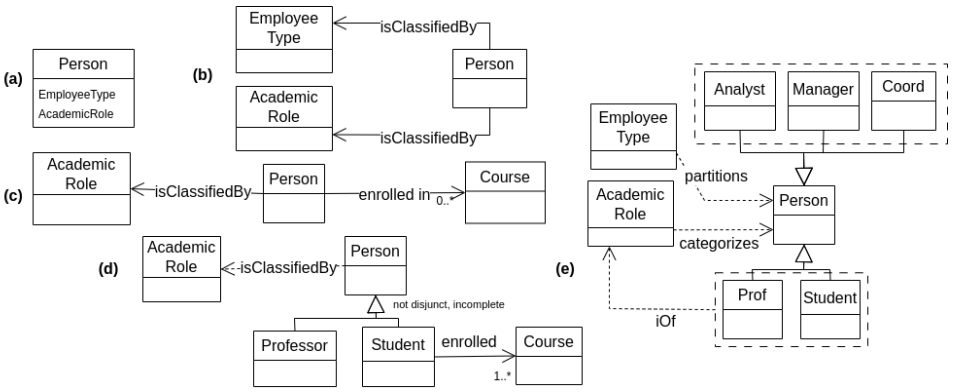

Figure 3: First steps applied to an example.

4.2. Modeling Type-Level Attributes

In addition to the initial steps described in Section 4.1, we present other cases in which the *Powertype* pattern can improve modeling by representing attributes in higher-order types.It is not rare to find models where the *basetype* and the *powertype* are collapsed into a single type, mixing their attributes. A typical modeling case is presented in Figure 4(a), where all instances of T_1 are assigned the same value for attribute $regAttr3$, while a different value is assigned to it in all instances of T_2. This kind of

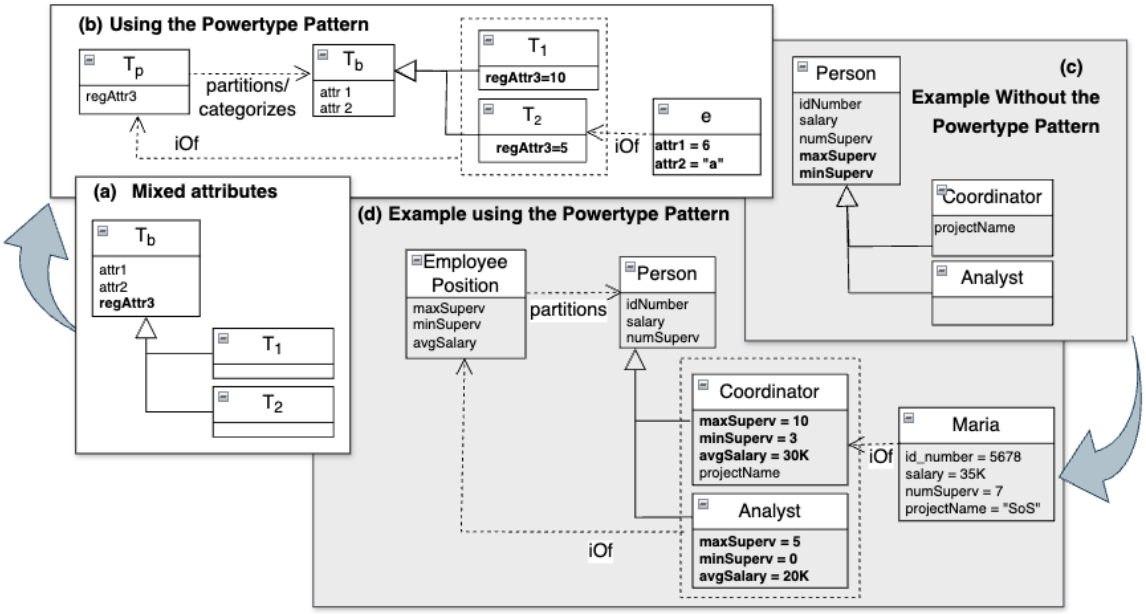

Figure 4: Modeling type-level attributes.

attributes have been called *regularity attributes* [9]. They aim at capturing regularities over lower-level type instances (i.e., instances of T_b subtypes), and constraining them, acting as patterns or defining intervals that must be obeyed.

Thus, we propose **Step 4**, as a good practice in this case: create the *powertype* T_p, as shown in Figure 4(b), and move the $regAttr3$ attribute to T_p. Thus, this attribute will be assigned values in instances of T_p (T_1 and T_2), constraining instances of T_1 and T_2 (e). Also, it may be useful to express patterns or rules that instances of T_b must comply with, e.g. $T_b.attr1 > T_p.regAttr3$. With this modeling step, these constraints are made explicit and allow us to formulate the following CQ:

CQ5: What are the value restrictions that an attribute $attr$ of a type T_i should respect for all instances $e \in T_i$?

Figure 4(c) illustrates this step using a domain-specific model. In this example, we have the type *Person* corresponding to T_b from Figure 4(a). *Person* subtypes share attributes such as *minSuperv* and *maxSuperv*, which are regularity attributes. Indeed, in all instances of the type $Coordinator$, the $minSuperv$ attribute will be assigned 3, while in all instances of the type $Analyst$, it would be assigned 0. After identifying these regularity attributes, the type *Employee Position* (*powertype(Person)*) is created, and, the regularity attributes are moved to the new type, as shown in Figure 4(d). With this, type $Person$ instances are now constrained according to the *Employee Position* type. Moreover, *Person.numSuperv* attribute values must obey the type *Employee Position*. In the example, "Maria" is an instance of *Person* whose *numSuperv* attribute is set to 7, respecting the *Coordinator* type interval constraint ($maxSuperv = 10 \wedge minSuperv = 3$).

Finally, some attributes may be derived from the instances of T_b, which are known as *resultant* attributes [9]. These attributes are usually numeric and calculated based on the subtype instances (e.g. average, rate, percentage, etc.), and thus they should belong to the higher-order type T_p. In the example of Figures 4(c)-(d), note that type *Person* has *salary* attribute that will have values for all its instances. This may be an opportunity to create the *resultant* attribute *avgSalary* at the *Employee Position* type, and maintain its values based on the values of the *salary* attribute.

After applying this step, it becomes possible to formulate and answer the following CQs:

- What is the average salary of a Coordinator?

- What is the minimum and maximum number of supervisions that are required to become a Coordinator?

4.3. Modeling Subtype Restrictions

In addition to the steps outlined in the previous sections, when modeling multiple levels, the modeler should be aware of the relationships that connect subtypes. It might be useful to represent these relations at the *Powertype* level. Suppose, for instance, that there is a relationship named *connectsTo* between T_b subtypes, T_1 and T_2, as shown in Figure 5(a), meaning that it connects instances of T_1 to instances of T_2. However, once the *Powertype* T_p has subtypes of T_b as its instances, this rule can be expressed by a self-relationship named *mayConnectTo* at the *Powertype*, as shown in Figure 5(a). Thus, **Step 5** consists of expressing the subtype relationships as a rule at a higher-order level type (T_p and its self-relationship), which allows the following CQ to be formulated and answered:

CQ6: Which instances of T_b could be connected to each other? In other words, given an instance of T_b, to which other instances could it be connected?

Furthermore, let us consider that the *basetype* T_b may have many subtypes, forming a flat tree with many siblings in a sequence. Now, suppose the *connectsTo* relationship occurs between many of these sibling pairs (e.g. (T_1,T_2), (T_2,T_3),...,(T_{n-1},T_n)). Although these relationships have similar semantics, in principle, it is not possible to generalize them as each one connects instances of specific subtype pairs. However, once the connection rule is preserved through the relationship *mayConnectTo* at type T_p, it becomes possible to generalize those relationships, substituting them by a single self-relationship at T_b. Therefore, with the help of the *Powertype* pattern, **Step 6** simplifies the model by reducing the number of relationships, as shown in Figure 5(b).

Figure 5(c) shows a domain-specific example where it is necessary to represent relationships between two pairs of subtypes in the hierarchical structure. The *analystRespondsTo* relationship specifically connects instances of types *Analyst* and *Coordinator*, while the *coordinatorRespondsTo* connects instances of types *Coordinator* and *Manager*.

Applying the steps described before, the model evolves to the one in Figure 5(d). It leverages the *Powertype* pattern to represent only two relationships, a self-relationship in the *basetype Person* and another in the *powertype Employee Position*. With this remodeling, it becomes possible to answer the following CQ: *Which instances of type Person can respond to another?* In other words, which restrictions should be applied to the instantiation of the *basetypeRespondsTo* relationship between *Person* instances? According to the *powertype Employee Position* and its instances, the answer is: *Analysts* can respond to *Coordinators*, and *Coordinators* can respond to *Managers*. It is worth noting that steps 5 and 6 can be applied recursively at each level of a higher hierarchical structure.

5. Reviewing The MiScOn Ontology

This case study addresses multi-level modeling challenges in the development of the MiScOn ontology (Military Scenario Ontology) [20], in light of the steps and competency questions discussed in Section 4. MiScOn captures both tactical and communication system aspects of military operations, grounded in military doctrines and validated by domain experts. It was chosen because it was developed based on the SABiO methodology [21], which distinguishes clear stages of the development process and their respective output artifacts. First, a reference ontology is generated and, later, the corresponding operational version is generated.

Moreover, OntoUML was used as MiScOn modeling language. While reviewing the MiScOn reference artifact, this allowed the modeler not only to ground the modeling on the Unified Foundational Ontology (UFO) [22], but also to count on the support of the Visual Paradigm OntoUML plugin[3], which was used to generate the MiScOn operational artifact.

[3]https://github.com/OntoUML/ontouml-vp-plugin

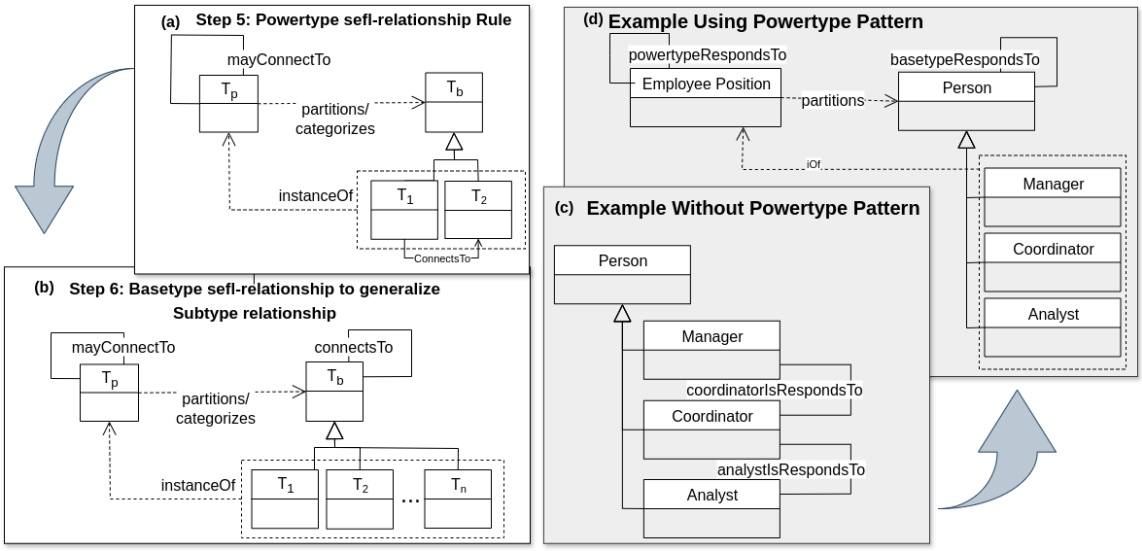

Figure 5: Modeling subtypes restrictions through the *Powertype* pattern.

5.1. Reference Ontology

During the process of reviewing the MiScOn ontology, the modeling of hierarchical structures in military organizations was refactored. The OntoUML extension proposed by Fonseca et al. [23] incorporated the MLT concepts and was therefore used in the reviewing of MiScOn to allow the representation of higher-order types and, at the same time, preserve the identity and temporal persistence of the entities.

The initial modeling of the structure of military organizations (MO) is shown in Figure 6(a). Note that MOs are classified according to several military organizational echelons, each with its own characteristics and relationships. For example, the 30^{th} Battalion, the 32^{nd} Battalion, and the 34^{th} Battalion are instances of the subtype *Battalion*, and are all commanded by the 11^{th} Brigade. Additionally, Battalions are commanded[4] by Brigades, Companies to Battalions, and so on. Moreover, each military organization has its own force (*strength* attribute), which represents the number of combatants it has.

The representation of organizational echelons in the military domain can benefit from the use of multi-level modeling seen in Sections 4.1 to 4.2. Promoting MO subtypes to *powertype* instances enables the definition of rules between these subtypes, which can be used to constrain MO instances. Additionally, it avoids the need for distinct relationships between each pair of the subtypes (subkinds), as presented by the authors in [24]. As seen in Figure 6(b), the model expresses the *isCommandedBy* relation, both at the *powertype* and at the *basetype*. In short, using the *Powertype* pattern simplified the model in Figure 6(a). Moreover, type-level attributes represented at the MO type were promoted to the *powertype* level as restrictions over the MO instances.

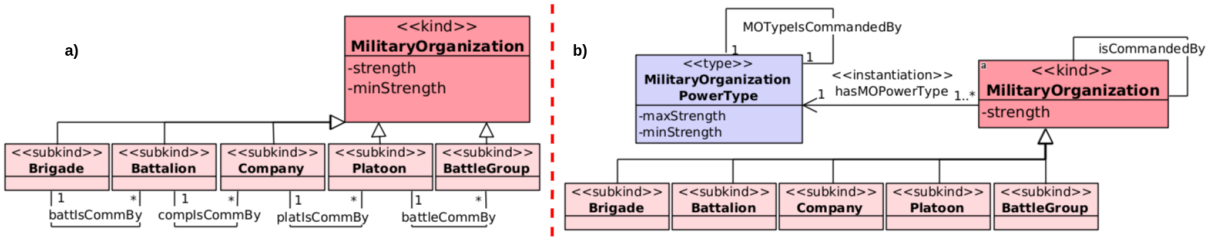

Figure 6: Fragment of the MiScOn reference ontology representing Military Organizations.

[4]It is important to highlight that the subordination relationship (isCommandedBy) in Figure 6 differs from the subordination concept of MLT (A is subordinate to B iff the instances of A are subtypes of instances of B), because it is related to the subordination relation in the military hierarchy.

Using the *powertype* `MilitaryOrganizationPowertype` (MOPT) and the basetype `MilitaryOrganization` (MO), it becomes possible to answer some CQs. CQ2: MOPT is the only type that partitions MO, through the *hasMOPowerType* relation. The cardinality (1 - 1..*) guarantees that for each MO instance, there is an instance of exactly one instance of MOPT. Conversely, for each MOPT instance, there is at least one instance of MO. CQ4 can also be answered, since MO can be partitioned according to MOPT instances ("Brigade", "Battalion", etc.). Moreover, restrictions on relationships and attributes are made explicit at MOPT, answering CQ5 and CQ6, respectively. According to MOPT, a *Battalion* can be commanded by exactly one *Brigade*. For example, given the MO 36^{th} *Battalion*, which is an instance of type (*hasMOPowerType*) *Battalion*, it can be commanded by the 11^{th} *Brigade*, and its *strength* attribute can assume values between *maxStrength* and *minStrength* values of the MOPT *Battalion* instance.

CQ2, CQ4 and CQ6 can also be answered in the modeling fragment on military vehicles and their various types, each of which can have different attributes depending on their respective *powertypes*, as shown in Figure 7. The *Kind* `Vehicle` is specialized in `Armored Vehicle`, which has, in turn, various subtypes, such as `Guarani`, `Urutu` and `Cascavel`. With the *VehiclePowerType*, it was possible to represent the regularity attributes at the power type level, restricting the instances of the *Armored Vehicle* type concerning the minimum and maximum passenger capacity and the maximum speed on land, according to the corresponding instance type.

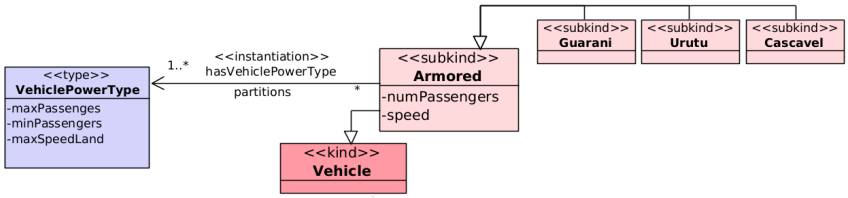

Figure 7: Fragment of the MiScOn reference ontology representing armored vehicles.

5.2. Operational Ontology

Following the development of the reference ontology in OntoUML, we implemented an operational ontology based on OWL language, incorporating instantiations and inference rules expressed in SWRL. To support multi-level modeling, we adopted the **punning** technique, which allows the same URI to be used as both a class and an individual. This enables reasoning across different levels of abstraction and supports rules that depend on treating entities as both types and instances. Furthermore, the lightweight version of UFO named gUFO has support to work with MLT.

```
<owl:Class rdf:about="https://example.com/miscon#Battalion">
        <rdfs:subClassOf rdf:resource="https://example.com/miscon#MilitaryOrganization"/>
</owl:Class>
<owl:NamedIndividual rdf:about="https://example.com/miscon#Battalion">
        <rdf:type rdf:resource="https://example.com/miscon/gufo#SubKind"/>
        <rdf:type rdf:resource="https://example.com/miscon/miscon#MilitaryOrganizationPowerType"/>
        <MOTypeIsCommandedBy rdf:resource="https://example.com/miscon/miscon#Brigade"/>
        <maxStrength rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">800</maxStrength>
        <minStrength rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">500</minStrength>
</owl:NamedIndividual>
```

Listing 1: Battalion being represented as a class and as an individual in the operational ontology in RDF/XML syntax.

This approach is aligned with Lenzerini's work [25], which explores meta-level queries in OWL2 QL using punning to enable querying both the structure and the instance level of an ontology. In our case, punning facilitated the operationalization of multi-level constructs, as illustrated in Listing 1, where the class `Battalion` is represented both as a class and an individual. The complete reference

and operational ontologies are available in the project site[5].

The use of punning technique along with SWRL rules help on creating instantiation restrictions in the operational ontology. Listing 2 shows a rule in SWRL which enforces the fact that for one instance of a military organization (o_1) to be commanded by another (o_2), its corresponding *powertypes* must also be commanded by each other. It can be seen in Figure 6(b) that the relationship `isCommandedBy` is at the *basetype* level, while the relationship `MOTypeIsCommandedBy` is at the *powertype* level. Once the *powertype* level relationships are instantiated, then the relationship instantiations at the *basetype* level will be constrained by the specified rule.

```
MilitaryOrganization(?o2) ^ MilitaryOrganization(?o1) ^
isCommandedBy(?o2, ?o1) ^
differentFrom(?o1, ?o2) ^
MilitaryOrganizationPowerType(?ompt1) ^ MilitaryOrganizationPowerType(?ompt2) ^
hasMOPowerType(?o1, ?ompt1) ^ hasMOPowerType(?o2, ?ompt2) ^
-> MOTypeIsCommandedBy(?ompt2, ?ompt1)
```

Listing 2: SWRL rule to express the restriction on the isCommandedBy relationship between MOs, based on their powertypes.

6. Final Considerations and Future Works

This work proposed a novel approach for identifying characteristics in conceptual models that indicate the need for multi-level modeling. Focusing on the Powertype pattern within the context of Multi-Level Theory (MLT), we introduced a step-by-step process to make implicit modeling elements—such as attributes and relationships—explicit. Based on this process, we developed a set of domain-independent competency questions that reinforce the benefits of identifying and applying the Powertype pattern in various modeling scenarios. The proposed set is not exhaustive and their development remains an open area of research.

Our findings demonstrate that the Powertype pattern, when applied in light of MLT, significantly enhances semantic modeling. Furthermore, the proposed guidelines shows promise for supporting the process of ontological unpacking in existing conceptual models, where the identification of patterns and anti-patterns and the knowledge about how and when apply them is essential for achieving well-founded representations aligned with foundational ontologies.

Acknowledgments

This research has been supported by CAPES ("Coordenação de Aperfeiçoamento de Pessoal de Nível Superior"), and funded by FINEP/DCT/FAPEB (ref.: 2904/20 contract No. 01.20.0272.00) under the “System of Systems of Command and Control” project.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly and DeepL Translate for grammar and spelling check. The authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

[1] C. M. Fonseca, J. P. A. Almeida, G. Guizzardi, V. A. Carvalho, Multi-level conceptual modeling: Theory, language and application, Data & Knowledge Engineering 134 (2021) 101894.

[2] J. Odell, Power types, J. Object Oriented Program. 7 (1994) 8–12.

[5] https://github.com/comp-ime-eb-br/S2C2-IME

[3] L. Cardelli, Structural subtyping and the notion of power type, in: Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 1988, pp. 70–79.
[4] G. Guizzardi, A. Botti Benevides, C. M. Fonseca, D. Porello, J. P. A. Almeida, T. Prince Sales, Ufo: Unified foundational ontology, Applied Ontology 17 (2022) 167–210. doi:10.3233/AO-210256.
[5] H. Herre, General formal ontology (gfo): A foundational ontology for conceptual modelling, in: Theory and applications of ontology: computer applications, Springer, 2010, pp. 297–345.
[6] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, et al., Dolce: a descriptive ontology for linguistic and cognitive engineering (2003). URL: https://hdl.handle.net/20.500.14243/196545.
[7] N. Guarino, Bfo and dolce: So far, so close..., COSMOS + TAXIS 4 (2017).
[8] G. Guizzardi, Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling, in: International Conference on Conceptual Modeling, Springer, 2014, pp. 13–27.
[9] G. Guizzardi, J. Almeida, N. Guarino, V. A. Carvalho, Towards an ontological analysis of powertypes, in: Proceedings of the Joint Ontology Workshops 2015 Episode 1: The Argentine Winter of Ontology co-located with the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015); Buenos Aires, Argentina, July 25-27, 2015, volume 1517, RWTH, 2015.
[10] N. Guarino, T. P. Sales, G. Guizzardi, Reification and truthmaking patterns, in: Conceptual Modeling: 37th International Conference, ER 2018, Xi'an, China, October 22–25, 2018, Proceedings 37, Springer, 2018, pp. 151–165.
[11] V. A. Carvalho, J. P. A. Almeida, Toward a well-founded theory for multi-level conceptual modeling, Software & Systems Modeling 17 (2018) 205–231.
[12] V. A. Carvalho, J. P. A. Almeida, C. M. Fonseca, G. Guizzardi, Extending the foundations of ontology-based conceptual modeling with a multi-level theory, in: Conceptual Modeling: 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings 34, Springer, 2015, pp. 119–133.
[13] V. A. Carvalho, J. P. A. Almeida, G. Guizzardi, Using a well-founded multi-level theory to support the analysis and representation of the powertype pattern in conceptual modeling, in: Advanced Information Systems Engineering: 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings 28, Springer, 2016, pp. 309–324.
[14] J. P. A. Almeida, V. A. Carvalho, F. Brasileiro, C. M. Fonseca, G. Guizzardi, Multi-level conceptual modeling: Theory and applications, in: Proceedings of the XI Seminar on Ontology Research in Brazil and II Doctoral and Masters Consortium on Ontologies, October 1st-3rd, 2018., volume 2228, CEUR-WS, São Paulo, Brazil, 2018, pp. 26–41.
[15] B. Neumayr, K. Grün, M. Schrefl, Multi-level domain modeling with m-objects and m-relationships, in: Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modeling-Volume 96, Citeseer, 2009, pp. 107–116.
[16] G. Guizzardi, J. P. A. Almeida, Stability patterns in ontology-driven conceptual modeling, in: Proceedings of the XIII Seminar on Ontology Research in Brazil and IV Doctoral and Masters Consortium on Ontologies (ONTOBRAS 2020), volume 2728, CEUR-WS, 2020, pp. 148–160.
[17] J. D. Lara, E. Guerra, J. S. Cuadrado, When and how to use multilevel modelling, ACM Transactions on Software Engineering and Methodology (TOSEM) 24 (2014) 1–46.
[18] T. Halpin, Subtyping revisited, CEUR Workshop Proceedings 365 (2012).
[19] G. Guizzardi, N. Guarino, Explanation, semantics, and ontology, Data Knowledge Engineering 153 (2024) 102325. URL: https://www.sciencedirect.com/science/article/pii/S0169023X24000491. doi:https://doi.org/10.1016/j.datak.2024.102325.
[20] A. M. Demori, An Ontology-Based Approach for Reproducing Military Operation Scenarios [Uma Abordagem Baseada em Ontologia para Reprodução de Cenários de Operações Militares], Master's dissertation, Military Institute of Engineering, Rio de Janeiro, Brazil, 2023.
[21] R. Falbo, Sabio: Systematic approach for building ontologies, in: Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering co-located with 8th International Conference on Formal Ontology in Information Systems (FOIS 2014), volume 1301, CEUR Workshop Proceedings, Rio de Janeiro, Brazil, 2014.

[22] G. Guizzardi, Ontological foundations for structural conceptual models, Ph.D. thesis, University of Twente, 2005.
[23] C. M. Fonseca, G. Guizzardi, J. P. A. Almeida, T. P. Sales, D. Porello, Incorporating types of types in ontology-driven conceptual modeling, in: International Conference on Conceptual Modeling, Springer, 2022, pp. 18–34.
[24] A. M. Demori, J. C. C. Tesolin, M. C. R. Cavalcanti, D. F. C. Moura, Supporting simulation of military communication systems using well-founded modeling., in: ONTOBRAS, 2022, pp. 73–84.
[25] M. Lenzerini, L. Lepore, A. Poggi, Metamodeling and metaquerying in owl 2 ql, Artificial Intelligence 292 (2021) 103432.